

# Welcome to Software Engineering (IT314)

## Lab 8 UML Diagrams

Kalgi Gandhi, Aman Sinha

Course Instructor: Prof. Jayprakash Lalchandani



DA-IICT

# Outline

- UML Diagrams
- Types of UML Diagrams
- Behavioral Diagrams
  - State Machine Diagram
  - Communication Diagram
  - Use Case Diagram
  - Activity Diagram
  - Sequence Diagram
  - Timing Diagram
  - Interaction Overview Diagram
- Election management system: An example
- VP Tool Demo

# UML Diagrams

- UML stands for Unified Modeling Language
- It's a rich language to model software solutions, application structures, system behavior and business processes
- These diagrams are further classified as:
  - Structure Diagrams
  - Behavioral Diagrams

# Types of UML Diagrams I

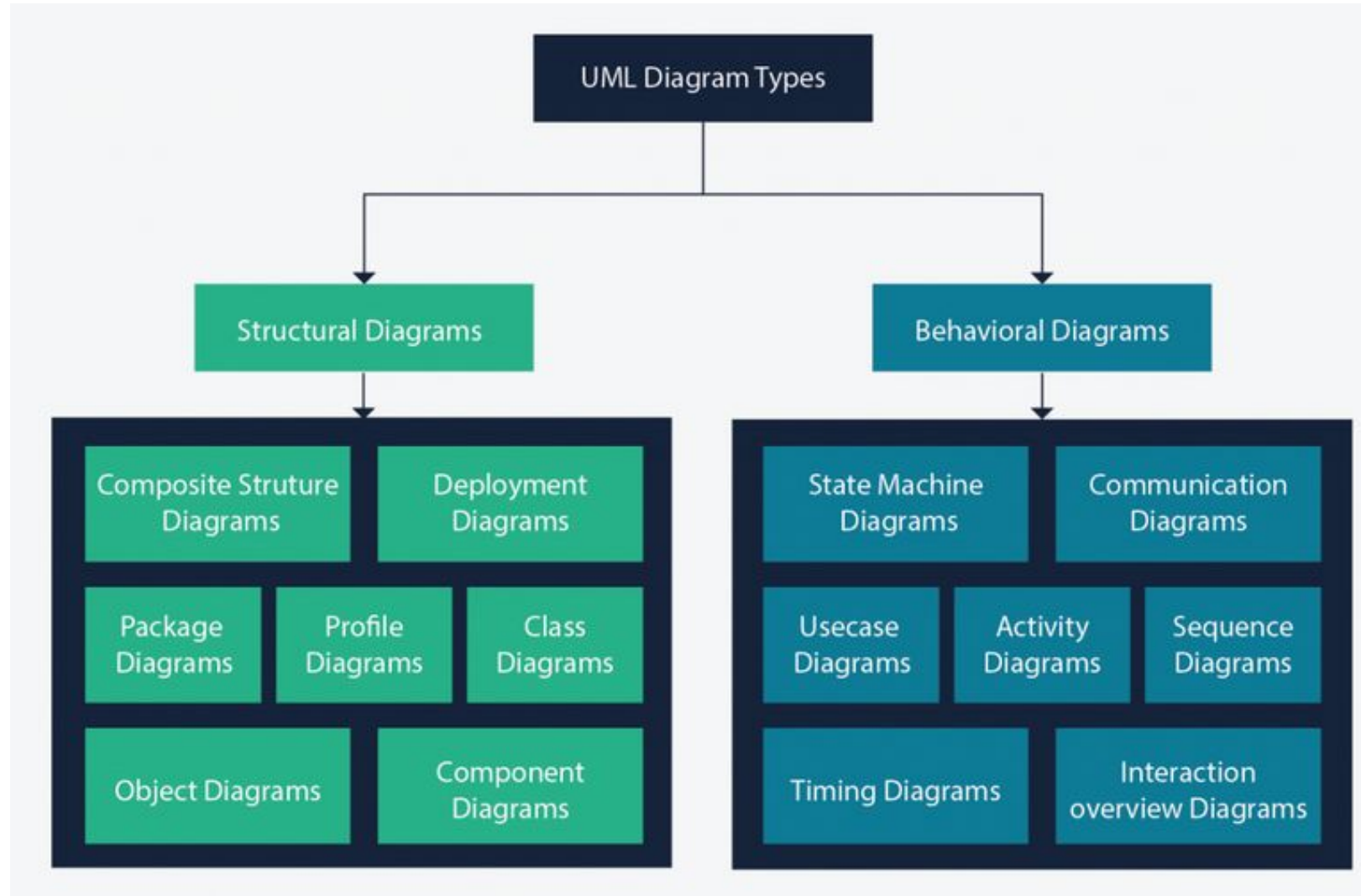
## Structure Diagrams

- Structure diagrams show the things in the modeled system
- In a more technical term, it shows different objects in a system

## Behavioral Diagrams

- Behavioral diagrams show what should happen in a system
- It describe how the objects interact with each other to create a functioning system

# Types of UML Diagrams II



# Behavioral Diagrams: Why to Study these???

- The objective of behavior modeling is to study/specify the behavior of the objects in a system
- At the logical level, the behavioral models allow us to:
  - Complete the structural model by finding the methods of our classes
  - Validate the structural model by making sure that all required attributes and associations are present
- At the physical level:
  - Sequence and Activity diagrams define a specification for our algorithms
  - State Machines can be used to generate executable code

# Sequence Diagrams

# Sequence Diagram

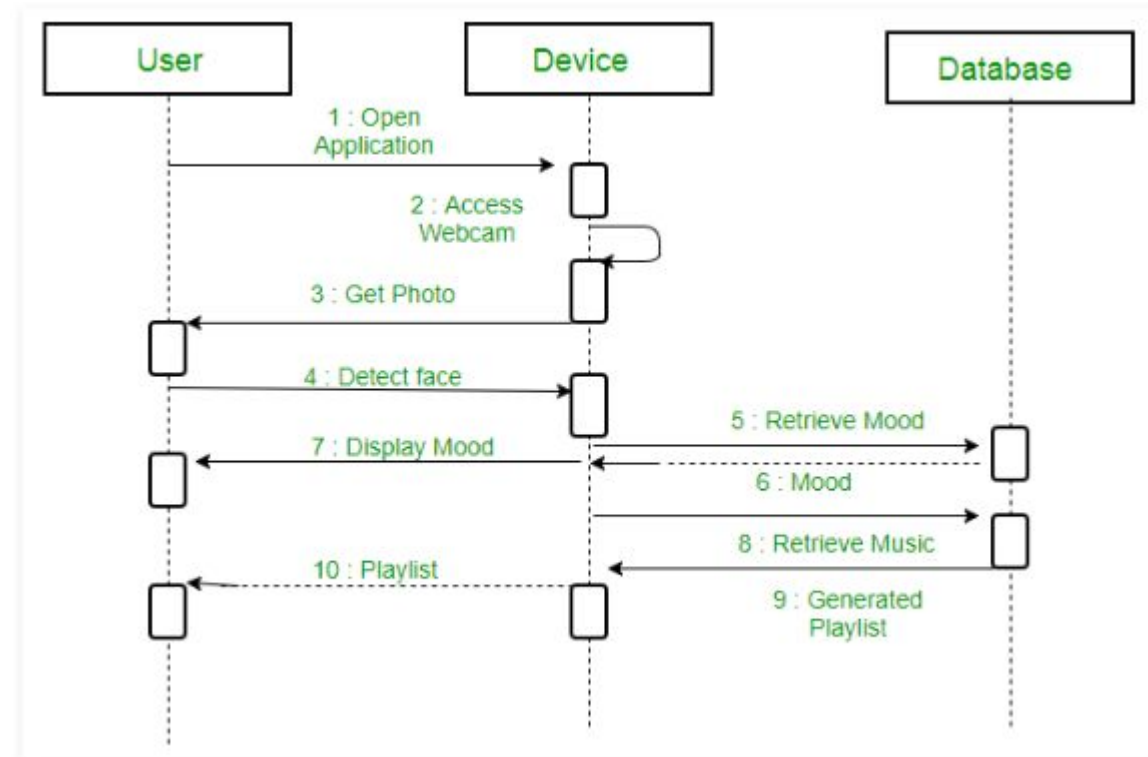
- Sequence diagram are also called as Event diagram or an Event scenario
- These are used to represent the order in which objects interact and how they interact, thereby enabling us to represent simple runtime scenarios
- This diagram basically represents the sequence of messages flowing from one object to another
- Importance of sequence diagram stems from the fact that interaction among the components of a system is very important for implementation and execution perspective
- Sequence diagram can be used to visualize the sequence of calls in a system to perform a specific task
- It is important to note that represents just an instance of a possible interaction and is not an exhaustive representation of the system's behavior
- In a sequence diagram processes are represented vertically and interactions among them are shown using arrows








# Sequence Diagram






There are two axes in a sequence diagram:

- The horizontal axis has the objects
- The vertical axis represents the progressing time, in the context of execution order









# Sequence Diagram: Important Symbols

To be represented	Symbol
Lifeline	
Actor	
Activity	
State	
Object flow	

To be represented	Symbol
Bars	
Initial State	
Control Flow	
Decision Activity	
Objects	

# Sequence Diagram: Important Symbols

To be represented	Symbol
Package	 An orange symbol representing a package, consisting of a large rectangle with a smaller rectangle attached to its top-left corner, resembling a folder.
Synchronous message	 A thick, dark gray horizontal bar representing a synchronous message.
Asynchronous message	 A thin blue horizontal arrow pointing to the left, representing an asynchronous message.

To be represented	Symbol
Create message	 A blue arrow pointing to the right with the text '<<Create>>' written above it.
Delete message	 A blue arrow pointing to the right with a large 'X' drawn over its tip, indicating deletion.
Self message	 A blue curved arrow that starts from a point, loops back, and ends with an arrowhead pointing back to the starting point, representing a self-message.

# Sequence Diagram: Important Concepts

- **Objects:**
  - Objects are the classes instances that participate in the interaction
  - Objects have lifeline
- **Actors:**
  - An actor in a UML diagram represents a type of role where it interacts with the system and its objects
  - An actor is always outside the scope of the system we aim to model
- **Lifelines:**
  - A lifeline is a named element which depicts an individual participant in a sequence diagram
  - Each instance in a sequence diagram is represented by a lifeline
  - Lifeline elements are located at the top in a sequence diagram
- **Difference between lifeline and actor:**
  - A lifeline always portrays an object internal to the system whereas actors are used to depict objects external to the system

# Sequence Diagram: Important Concepts

- **Messages:**

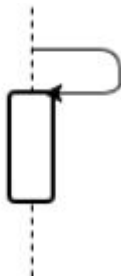
- Communication between objects is depicted using messages
- The messages appear in a sequential order on the lifeline
- Lifelines and messages form the core of a sequence diagram
- Types of messages:
  - **Synchronous messages:** A synchronous message waits for a reply before the interaction can move forward
  - **Asynchronous messages:** An asynchronous message does not wait for a reply from the receiver
  - **Create message:** We use a Create message to instantiate a new object in the sequence diagram
  - **Delete message:** We use a Delete Message to delete an object
  - **Self message:** Certain scenarios might arise where the object needs to send a message to itself. Such messages are called Self Messages
  - **Reply message:** Reply messages are used to show the message being sent from the receiver to the sender
  - **Found message:** A Found message is used to represent a scenario where an unknown source sends the message
  - **Lost message:** A Lost message is used to represent a scenario where the recipient is not known to the system

# Sequence Diagram: Important Concepts

- **Guards:**
  - Guards are used to model conditions in sequence diagrams
  - They are used when we need to restrict the flow of messages on the pretext of a condition being met
  - Guards play an important role in letting software developers know the constraints attached to a system or a particular process

# Sequence Diagram: Important Concepts

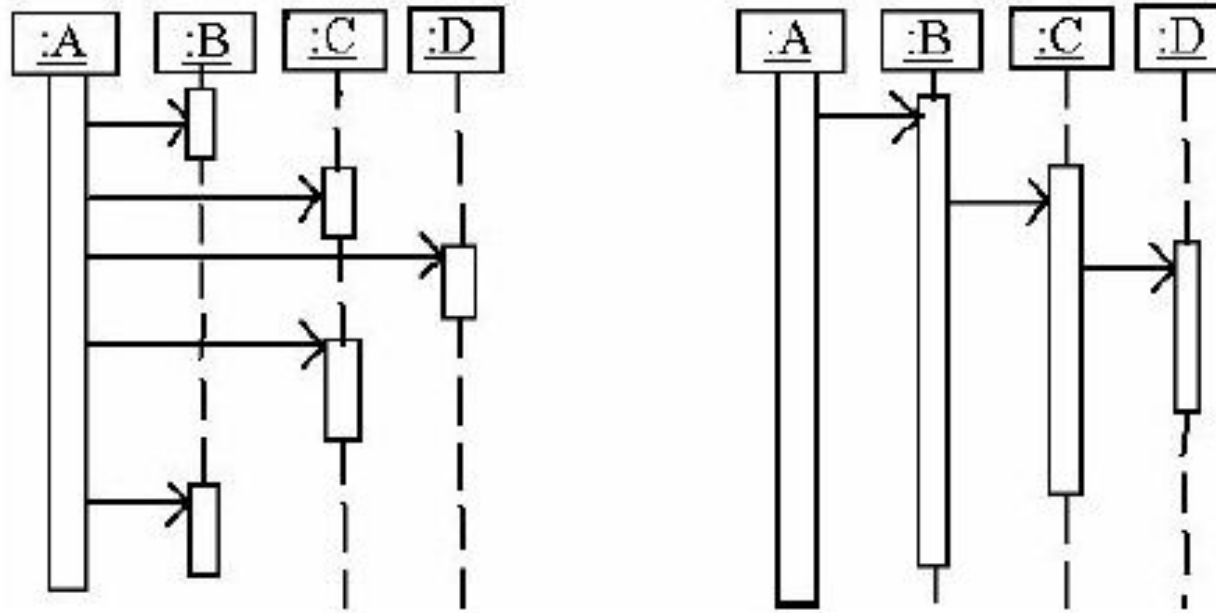
- **Activation:**
  - These are also called the method-invocation boxes
  - Activation elements in the UML sequence diagram are boxes on the lifelines
  - Indicate that an object is responding to a message
  - It starts when the message is received and ends when the object is done handling the message
- **Recursive calls:**
  - A call in which an object calls its own operations
  - Notation:



# Sequence Diagram: Important Concepts

- **Control flow types:**

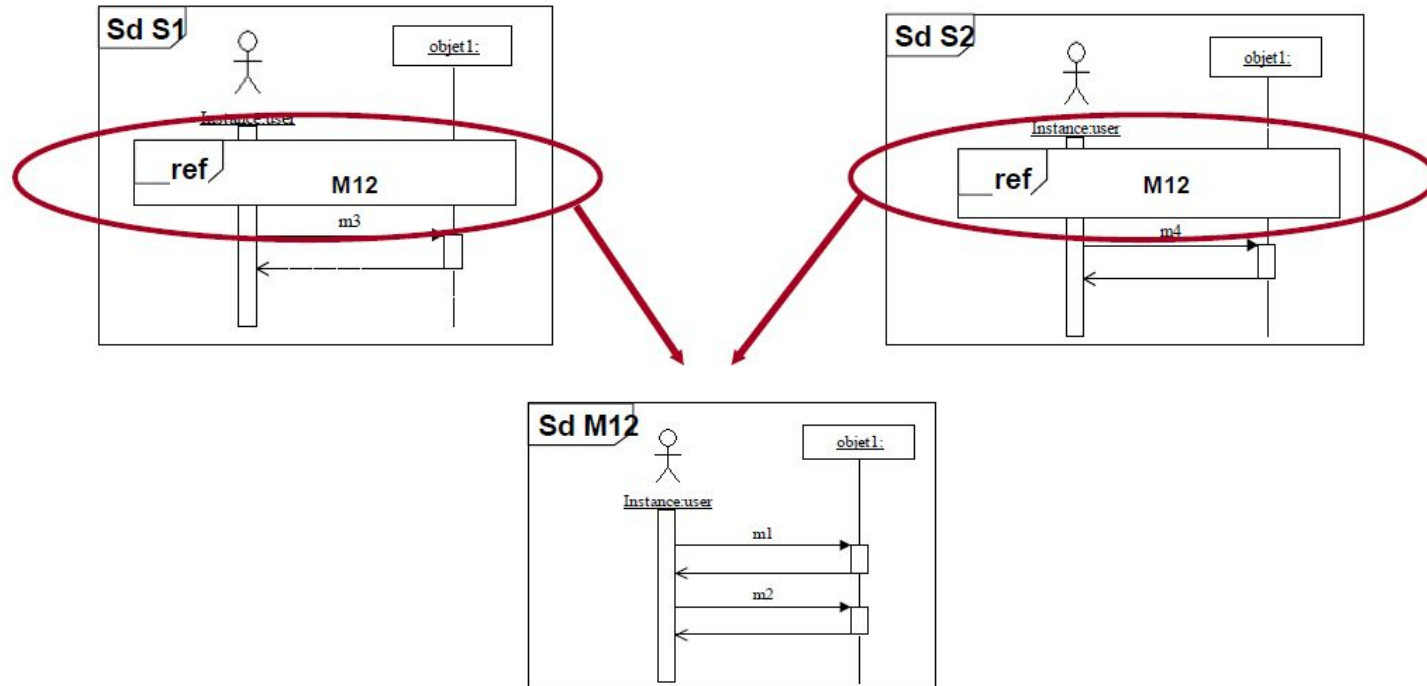
- Sequence diagrams can be used to easily and visually identify the application's control flow
- Control flow can be centralised or Distributed
- Example:



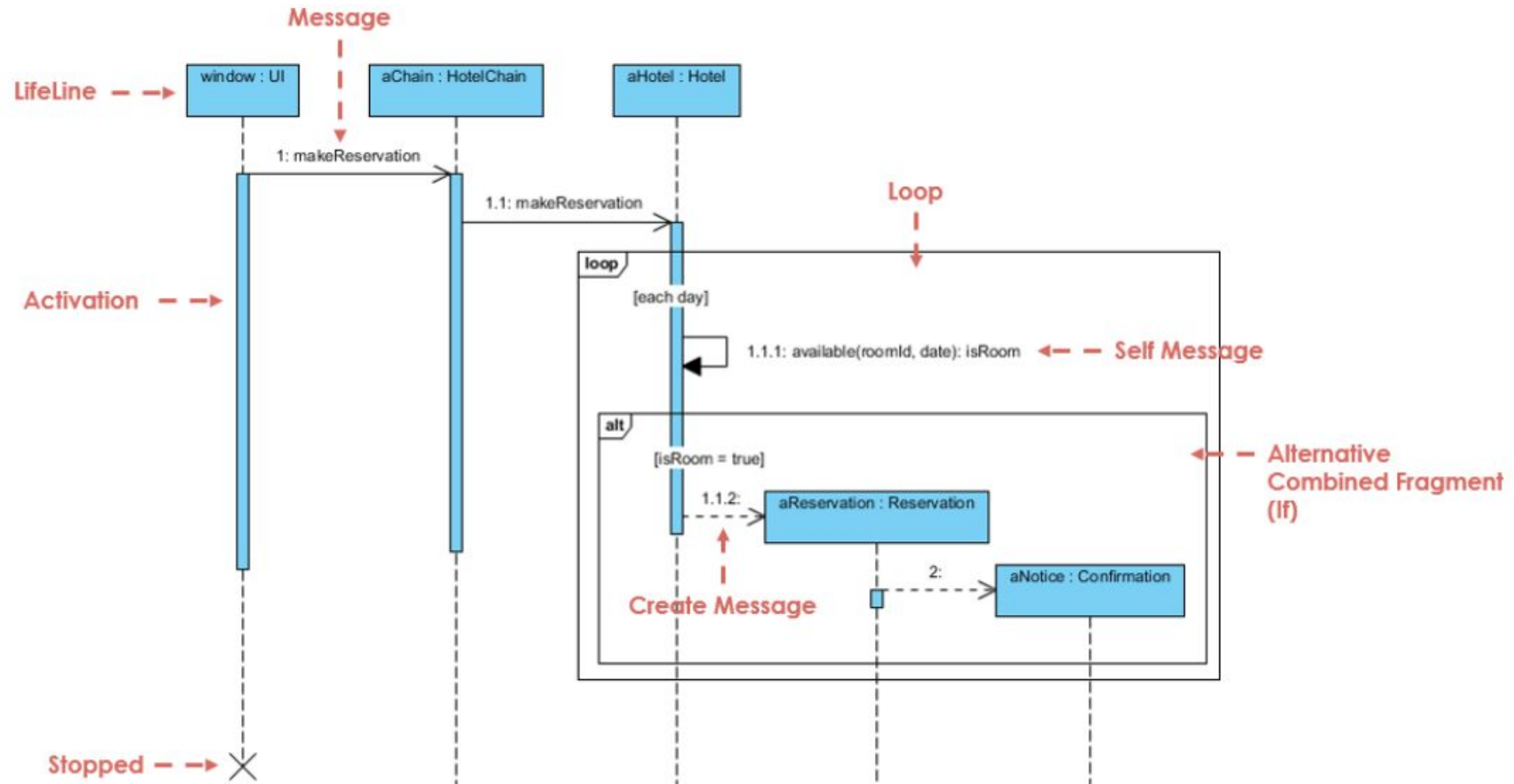


# Sequence Diagram: Important Concepts

- **Fragments:**
  - Fragments are used to represent:
    - Loops
    - Conditional branches
    - References to other sequence diagram, etc



# Sequence Diagram: An Example of Hotel Reservation

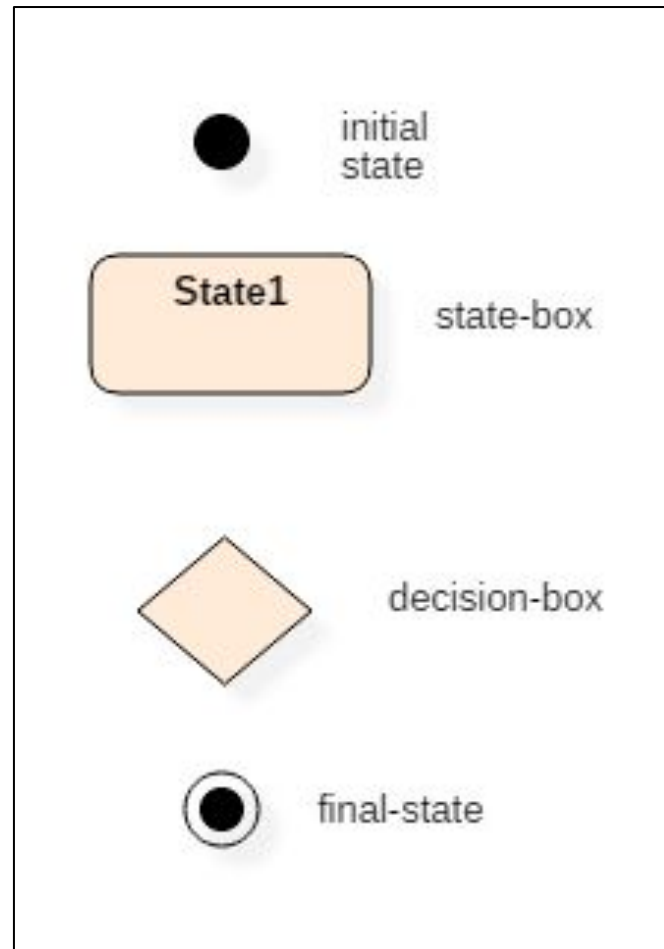


# State Machine Diagrams

# State Machine Diagram

- State machine diagrams are similar to activity diagrams, although notations and usage change a bit
- They are sometimes known as state diagrams or state chart diagrams as well
- These are very useful to describe the behavior of objects that act differently according to the state they are in at the moment
- The State machine diagram below shows the basic states and actions

# State Machine Diagram: Important Symbols



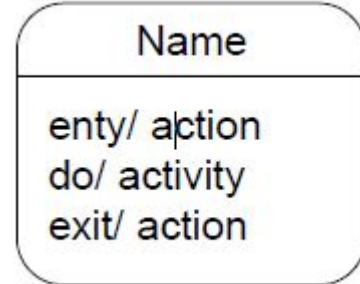
# State Machine Diagram: Important Concepts

- **State:**
  - A state is a stage of existence during which an object:
    - Satisfies certain conditions
    - Executes an activity or
    - Waits for an event
  - A state defines how the object reacts to new events

# State Machine Diagram: Important Concepts

- **Representation of state:**

- A state is described by a rounded rectangle containing:
  - **Name:** Must be unique within the class
  - **Entry and Exit actions:** Instantaneous processes executed by the object on entering/exiting the state
  - **Activity:** Long-lasting processes executed while the object is in that state
- Notation:



- **Final state:**

- A state diagram can have any number of final states and has no outgoing transitions
- The final state signifies that the object has terminated its execution, and has no further reason to exist

# State Machine Diagram: Important Concepts

- **Initial state:**
  - Any state diagram has exactly one initial state
  - The initial state is the point of creation of the object
  - The transition leaving it is taken when the object is instantiated
  - This transition may contain a guard condition and an action, but not an event
  - It may have multiple mutually-exclusive outgoing transitions, but no incoming transitions
- **Transition:**
  - A transition describes how an object moves from one state to another
  - Notation:

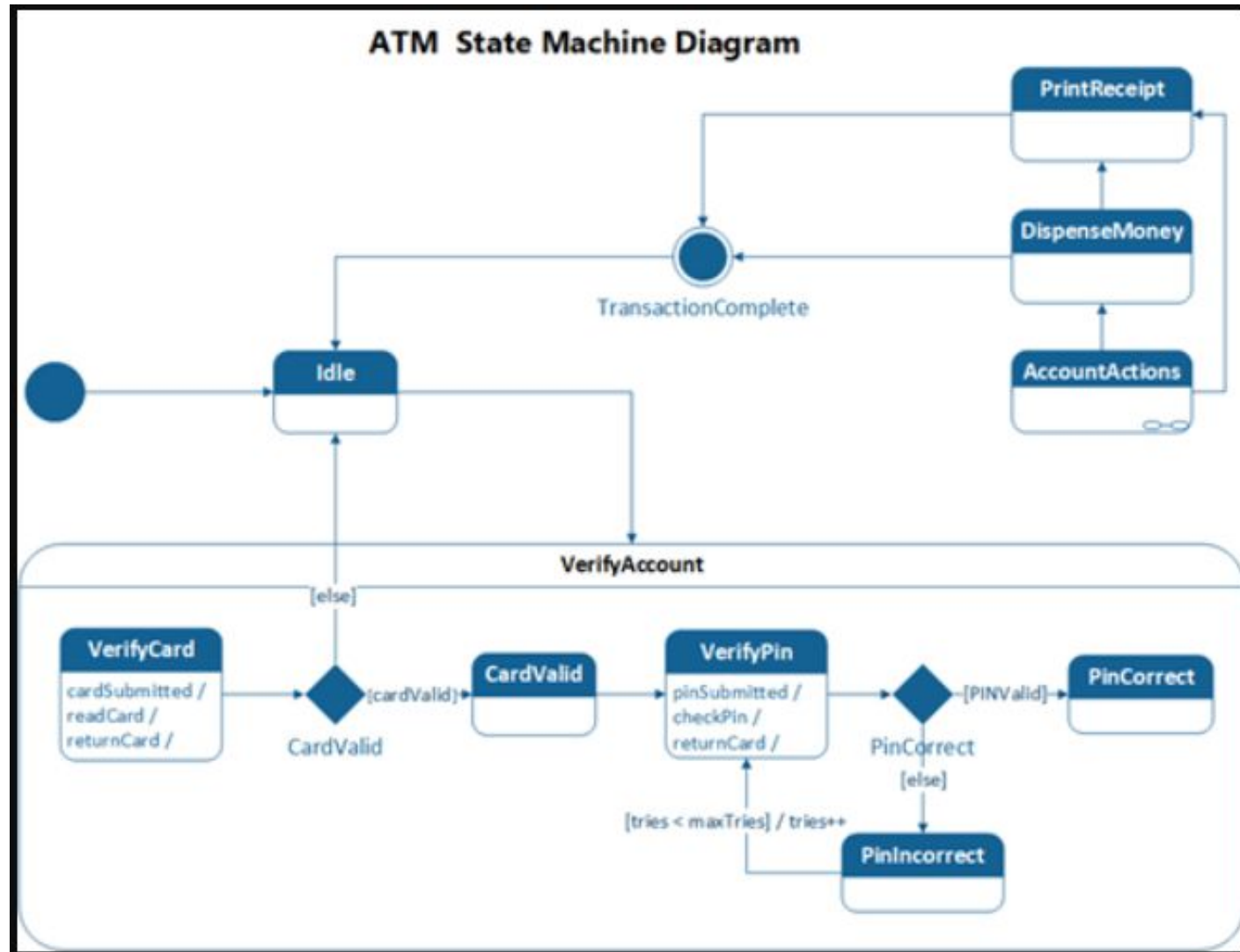
event [ guard condition ] / action



- The transition is taken when the event is received if the guard condition is true
- When the transition is taken, the action is executed by the object
- A transition without an event or guard condition is considered to be ‘automatic’



# State Machine Diagram: An Example of ATM



# Activity Diagrams

# Activity Diagram

- Activity diagrams are used to describe the flow of control in a system and consist of activities and links
- In other words, Activity Diagrams represent workflows in a graphical way and provide an understanding of how the system will work when executed
- The flow can be concurrent, sequential, or branched
- Activities are nothing but the functions of a system
- Many activity diagrams may be needed to capture the entire flow in a system
- These can be used to describe both business workflows and workflow of any component in a system
- Activity diagrams provide an important feature called as Swimlanes. They are used to represent ‘who is responsible for which activities being performed’
- In order to represent swimlanes, we split the Activity diagram into vertical sections, where each section corresponds to one actor







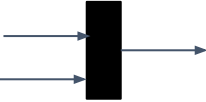
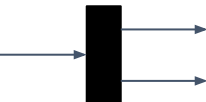
# Activity Diagram

- Used to model Workflow or Business Processes
- Very expressive to model Use Cases
  - Instead of sequence diagrams for instance
- More exhaustive and precise
- From UML 2.0, possible to model complex algorithms and operation's
- bodies
- Can be attached to:
  - A class
  - An operation
  - A use-case (workflow)

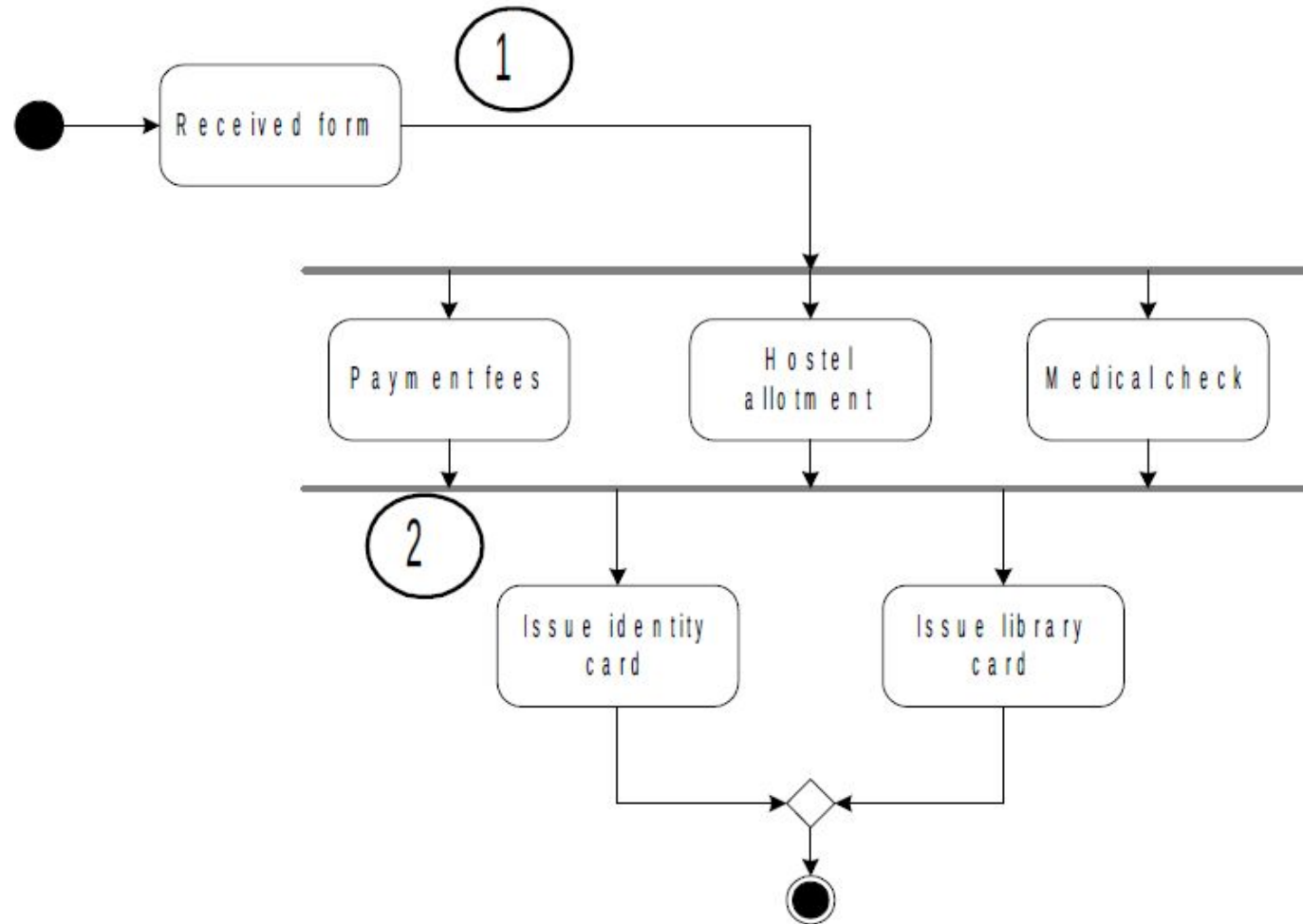
# Activity Diagram

- Since UML 2.0, we use the notion of action to designate atomic steps in a UML activity diagram (called activities in previous versions)
- An activity can be composed of:
  - Actions
  - Control Nodes (Fork, Merge, Decision, Join),
  - Object Nodes (input & output pins, object nodes, activity parameter node)
  - Arrows (Control Flow & Object Flow)
- The execution model of UML AD is based on the notion of production/consumption of token
  - Largely inspired from Petri Nets
- Transitions are automatic
- **Concurrent flow:**
  - Concurrent flows are used to model activities that can happen in parallel in the system
  - It is represented as a straight, slightly thicker line in an activity diagram
  - Fork Node and Join Node are used to represent concurrent flows

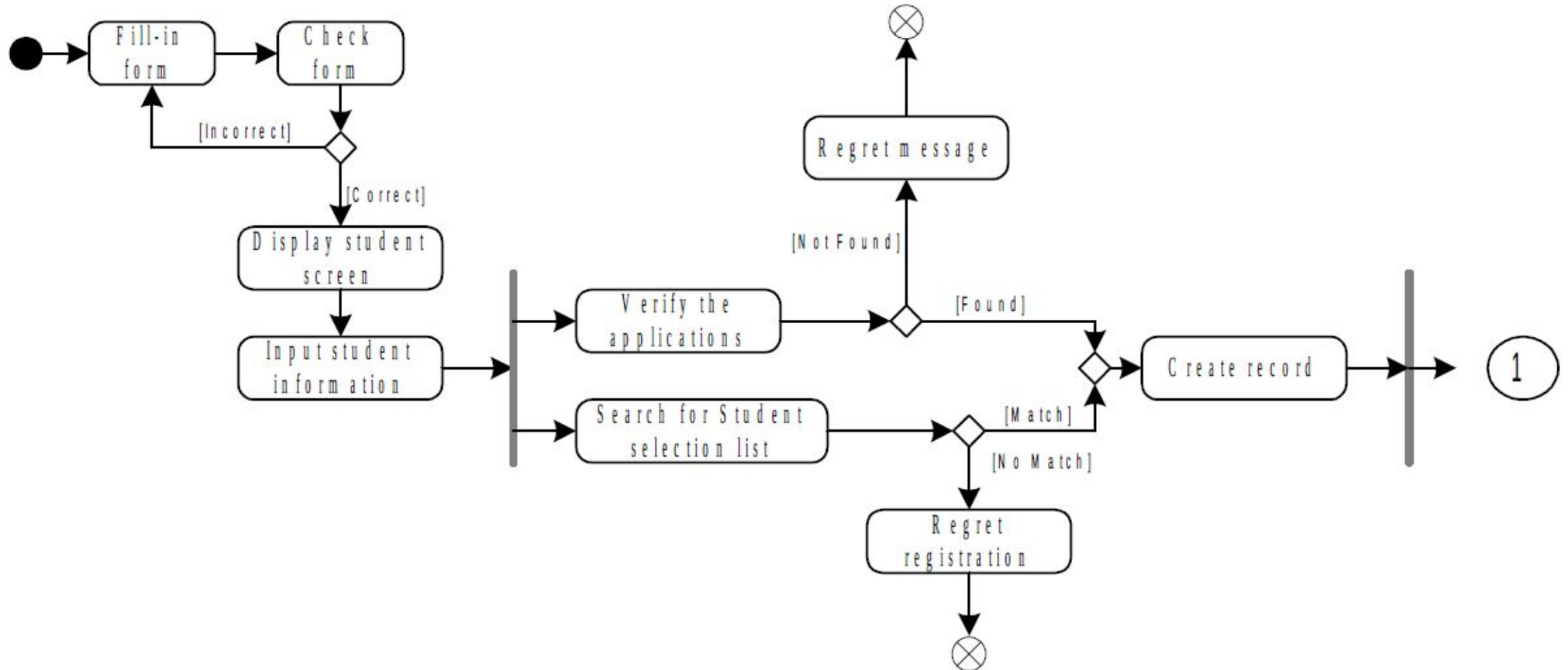
# Activity Diagram: Important Symbols

To be represented	Symbol
Initial state	
Final state	
Activity box	
Decision or branching box	
Action flow	
Action box	
Join Node	
Fork Node	

# Activity Diagram: An Example of Student Enrollment System

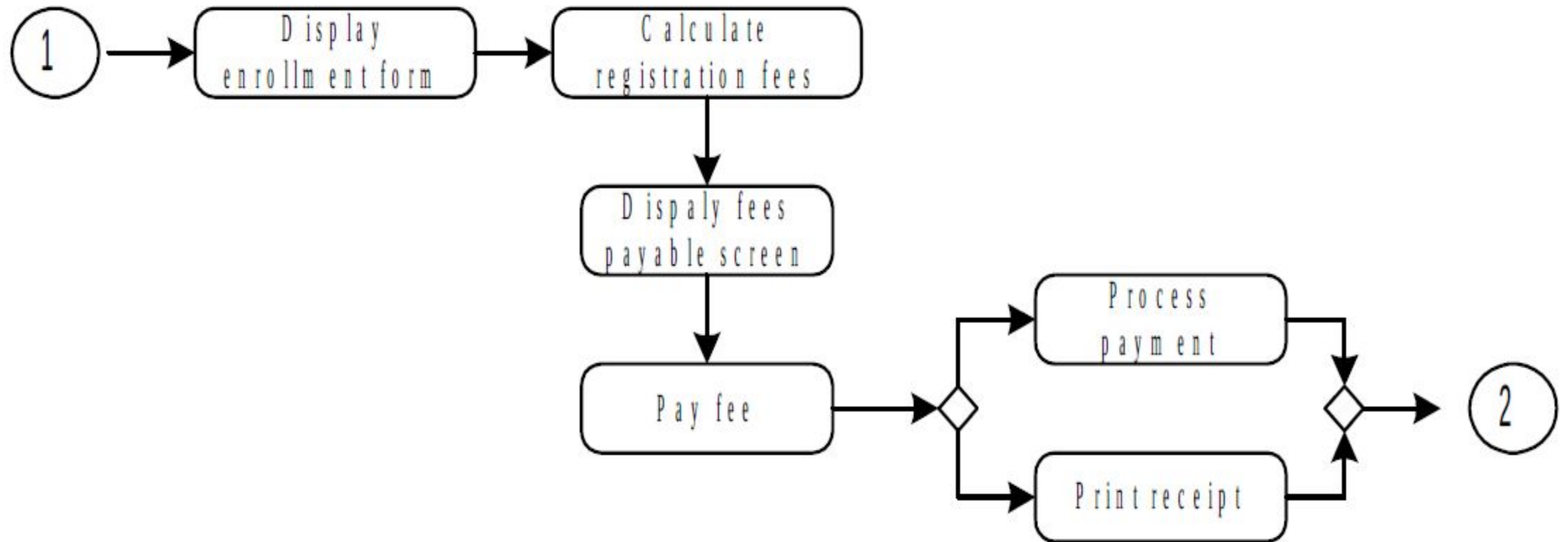


# Activity Diagram: An Example of Student Enrollment System

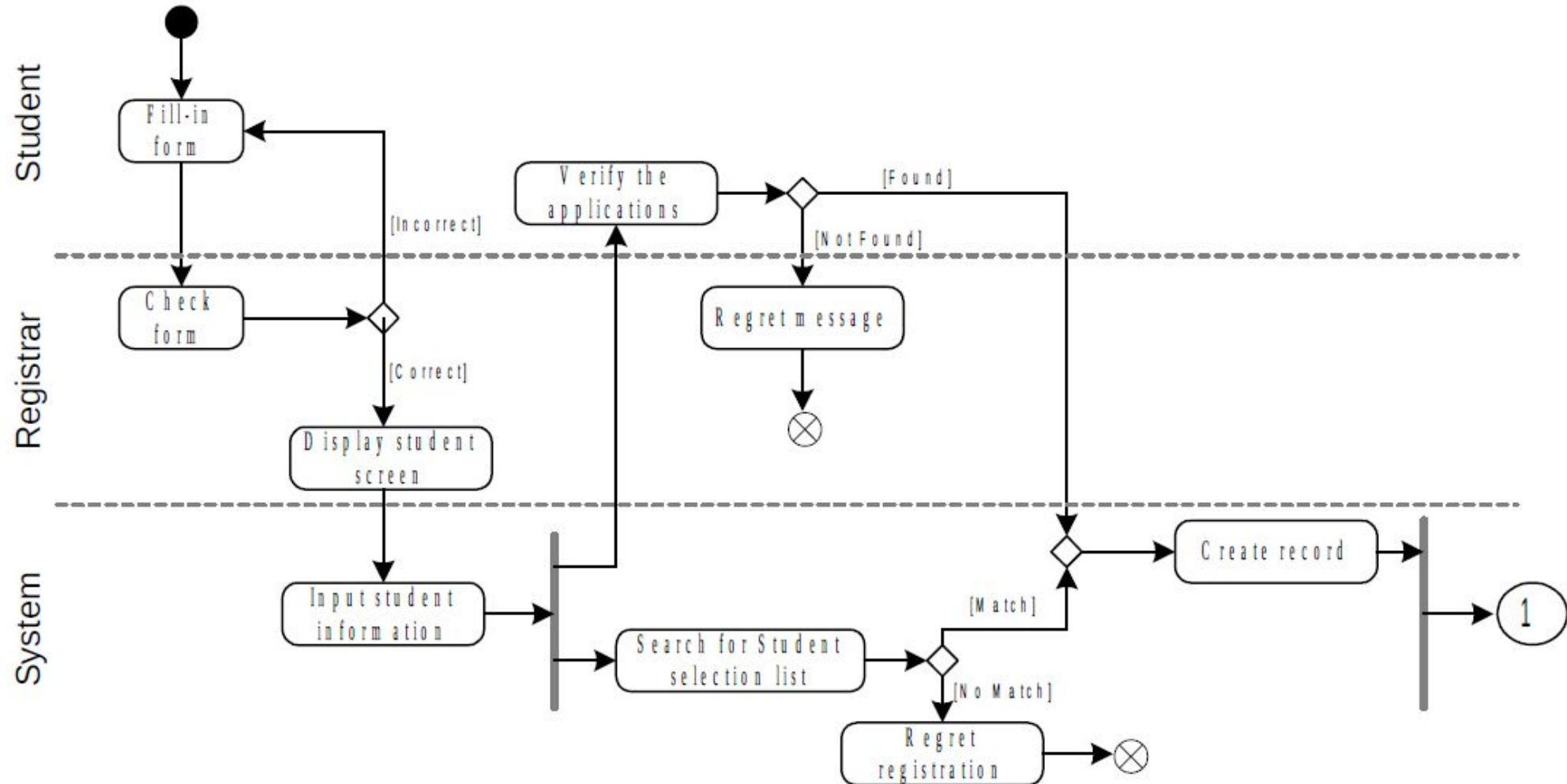




# Activity Diagram: An Example of Student Enrollment System



# Activity Diagram: An Example of Student Enrollment System



# Communication Diagrams

# Communication Diagram

- Also called collaboration diagrams
- Communication diagrams are similar to sequence diagrams, but the focus is on messages passed between objects
- The same information can be represented using a sequence diagram and different objects

## Why Communication diagram is needed?

- Communication diagrams are very useful for visualizing the relationship between objects collaborating to perform a particular task, however, this is difficult to determine from the sequence diagram
- Communication diagrams can also help to determine the accuracy of the static model (i.e. class diagram)

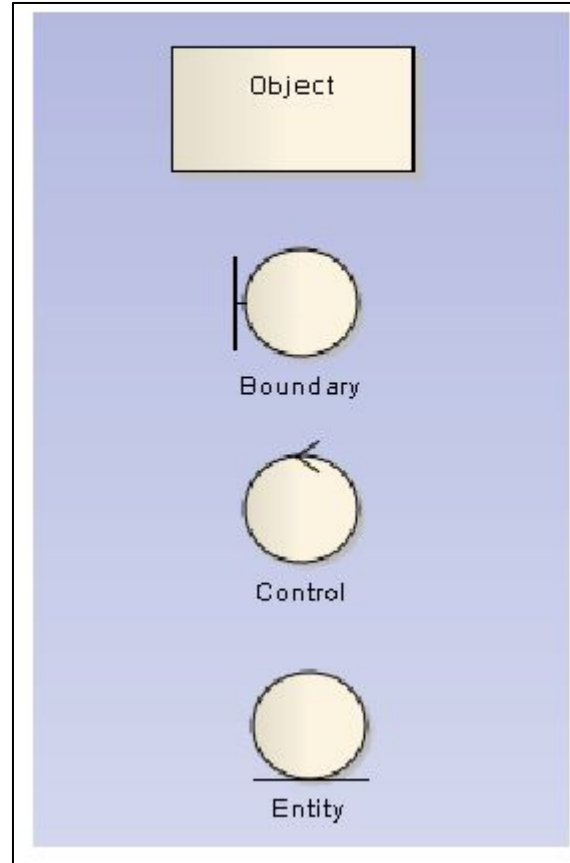
## Need of timeline in sequence diagram

- A sequence diagram is structured in such a way that it represents a timeline which begins at the top and descends gradually to mark the sequence of interactions

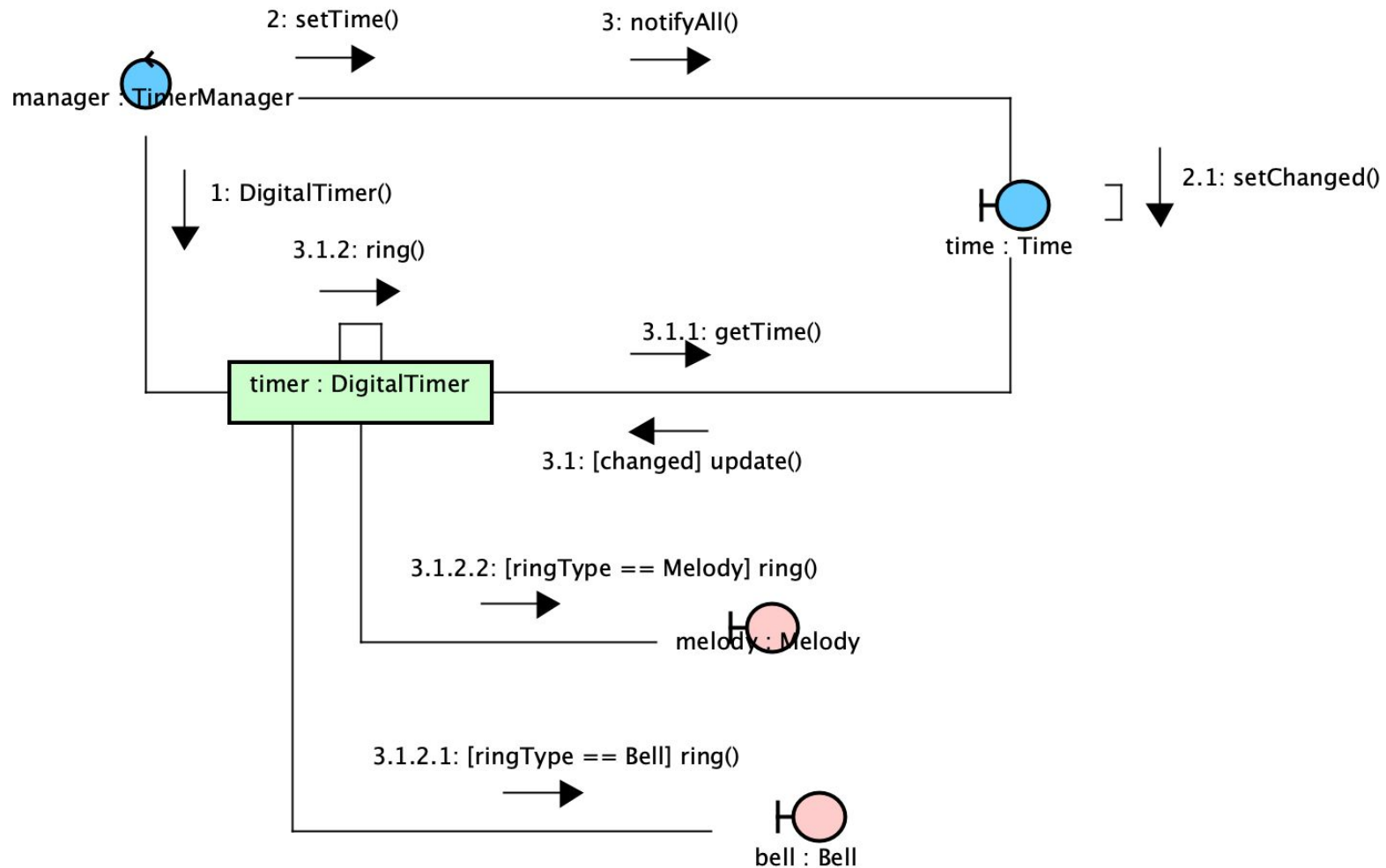
# Communication Diagram vs Sequence diagram

Sequence Diagrams	Collaboration Diagrams
The sequence diagram represents the UML, which is used to visualize the sequence of calls in a system that is used to perform a specific functionality.	The collaboration diagram also comes under the UML representation which is used to visualize the organization of the objects and their interaction.
The sequence diagram are used to represent the sequence of messages that are flowing from one object to another.	The collaboration diagram are used to represent the structural organization of the system and the messages that are sent and received.
The sequence diagram is used when time sequence is main focus.	The collaboration diagram is used when object organization is main focus.
The sequence diagrams are better suited of analysis activities.	The collaboration diagrams are better suited for depicting simpler interactions of the smaller number of objects.

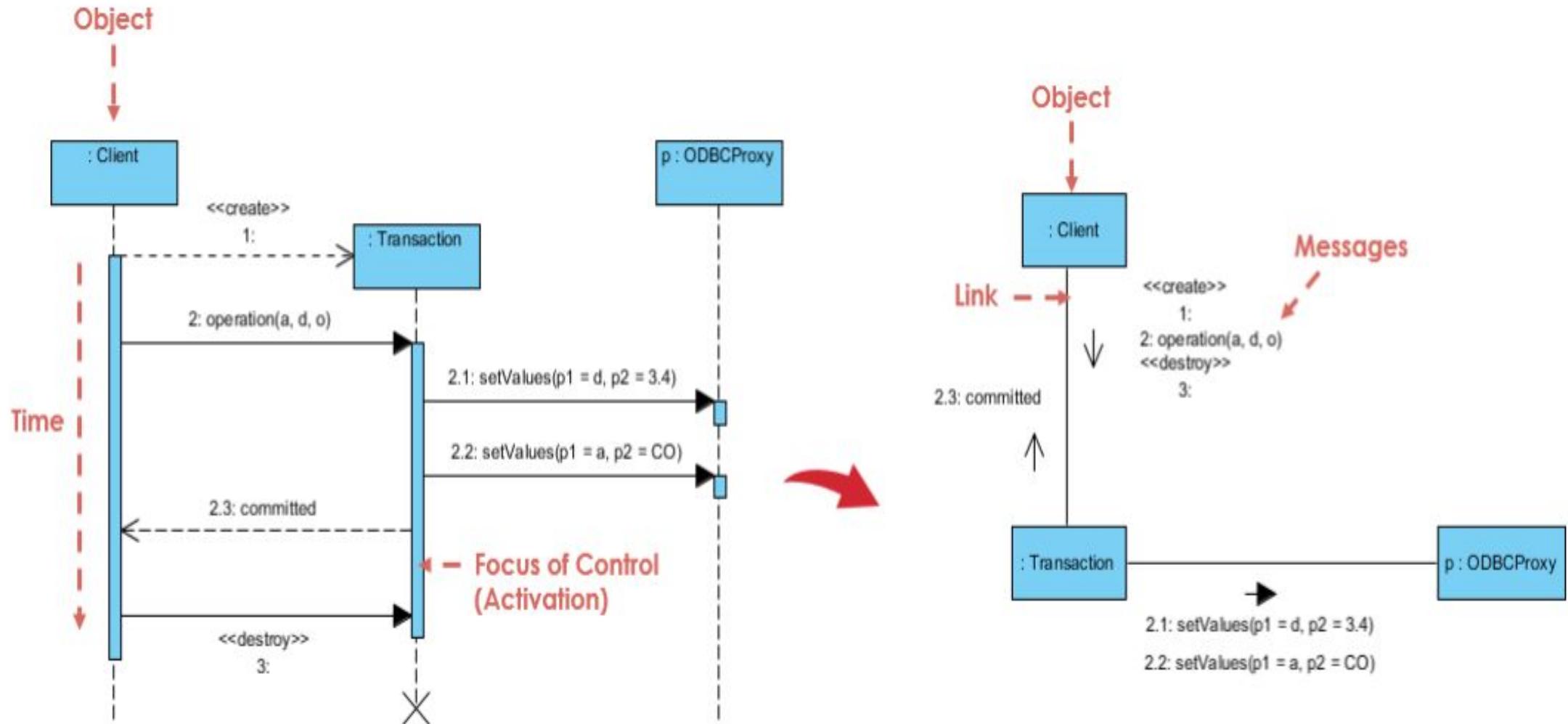
# Communication Diagram: Important Symbols



# Communication Diagram: An Example of Digital Timer



# Conversion Between Communication and Sequence Diagram: An Illustrated Example





# Use case Diagrams

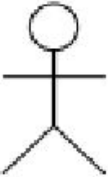


# Use Case Diagram

- Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact
- It's a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system

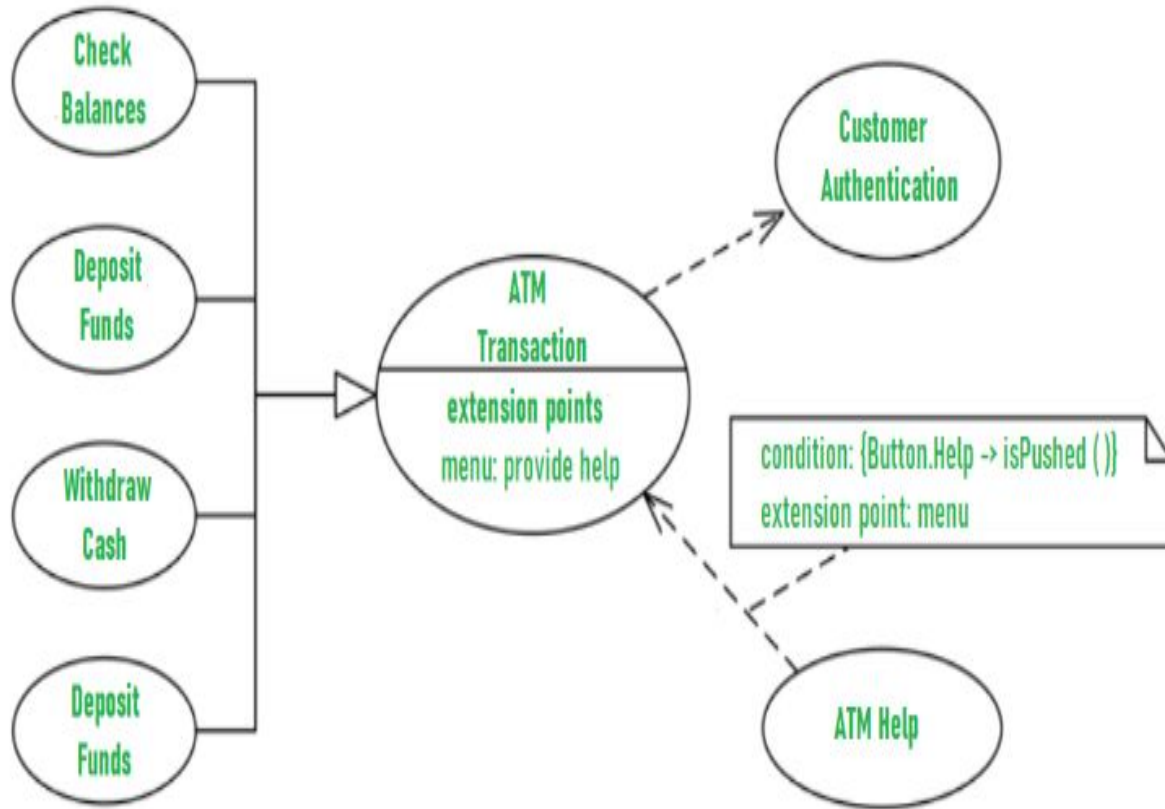
## Can there be multiple Use Case diagrams for a system?

- Yes, a system may have multiple use case diagrams
- The main purpose of having multiple use case diagrams for a system may be any of the following:
  - To showcase different viewpoints (ex: business vs system)
  - To showcase different levels of details
  - To showcase the different parts or components of a complex system to ensure easy readability and understanding

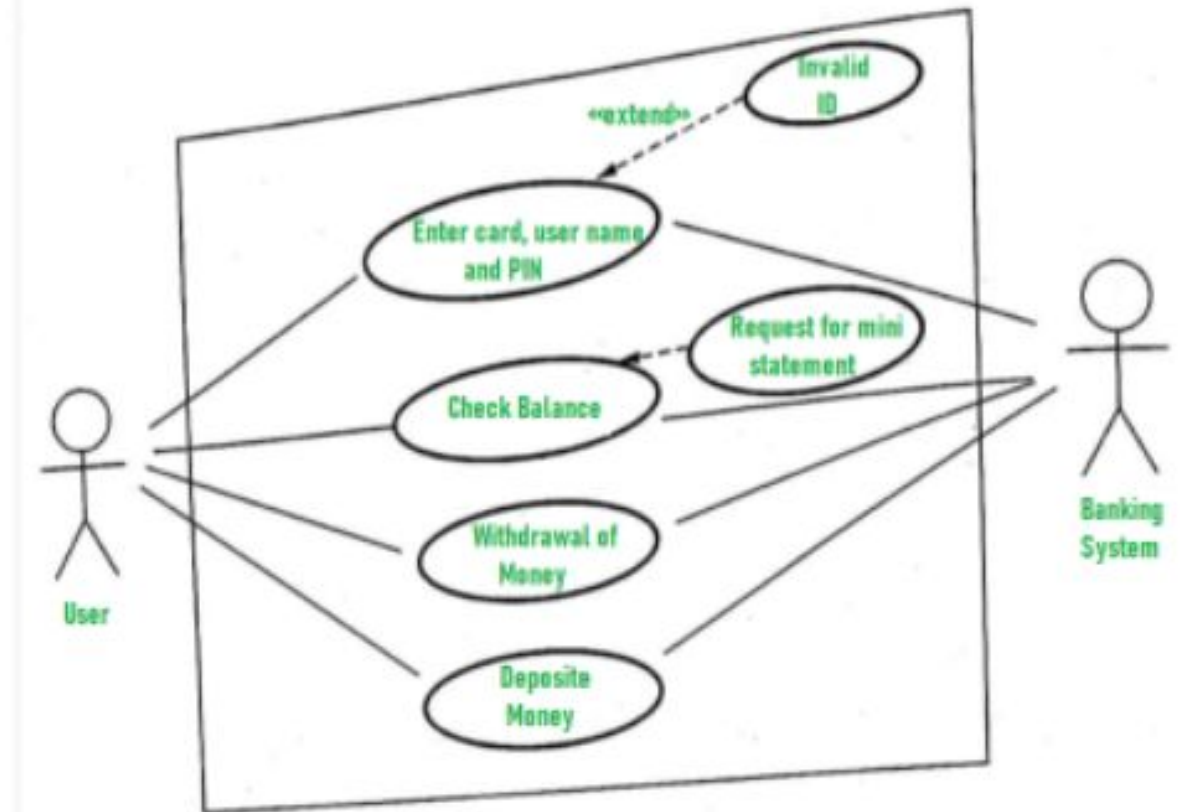
# Use Case Diagram: Important Symbols

Symbol	Reference Name
	Actor
	Use case
	Relationship

# Use Case Diagram: An Example of ATM - 1

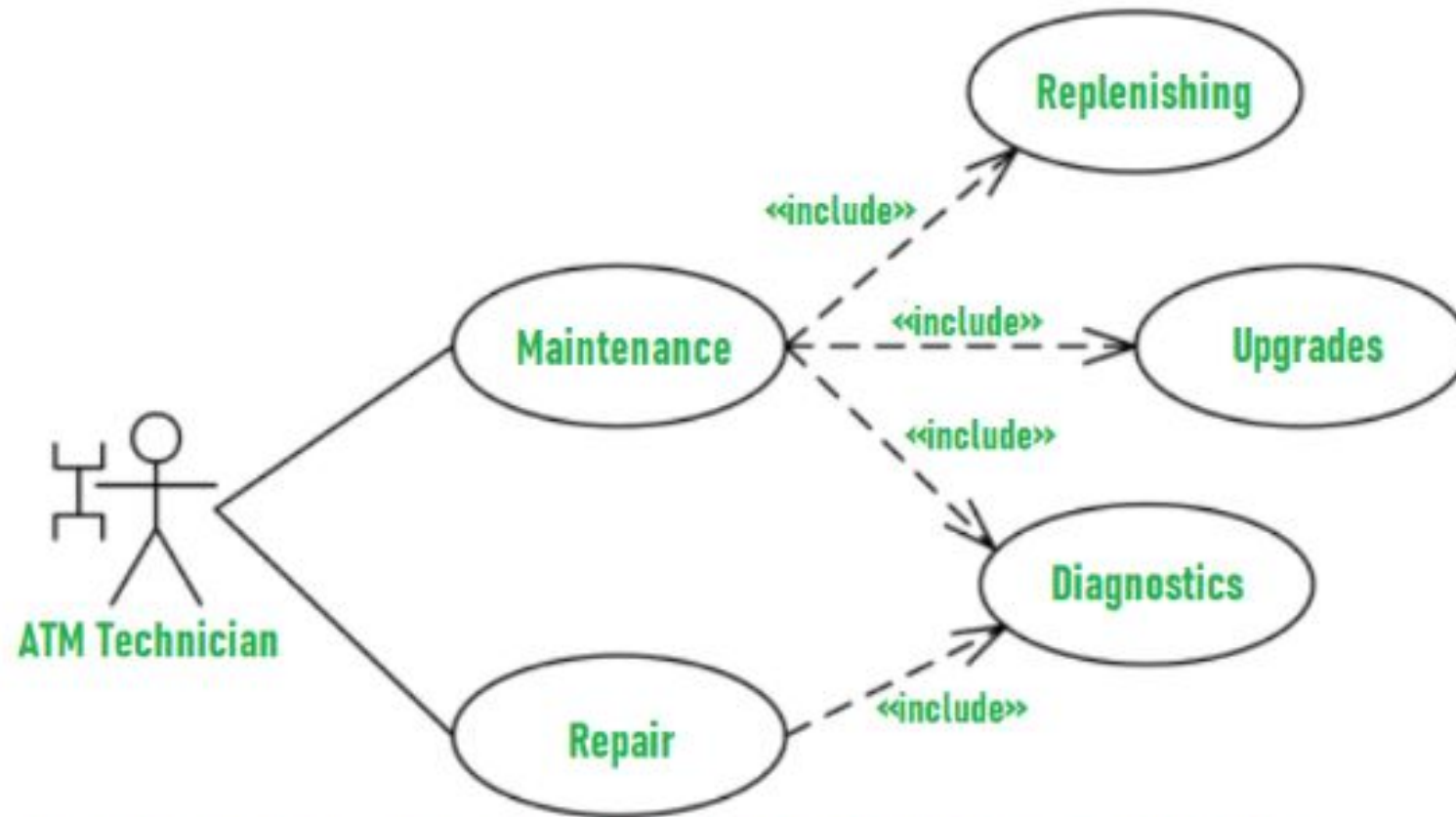


Use Case Diagram for Customer Authentication



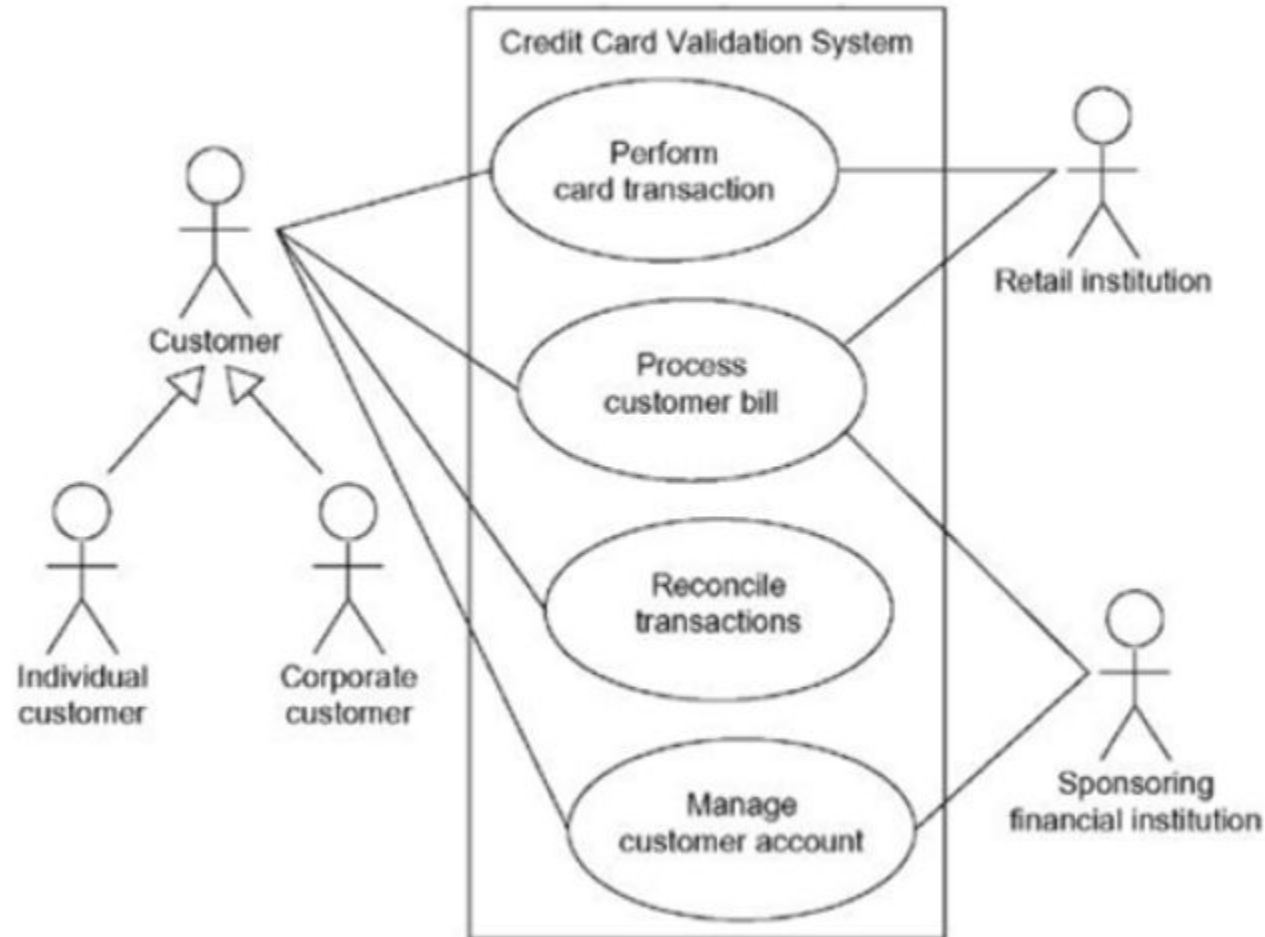
Use Case Diagram for Bank ATM System

# Use Case Diagram: An Example of ATM - 2



**Use Case Diagram for Working of ATM Technician**

# Use Case Diagram: An Example with hierarchy



# Timing Diagrams

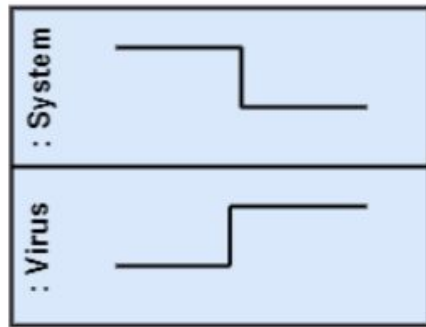
# Timing Diagram

- Timing diagrams are used to represent the behavior of objects in a given time frame
- Often described as an inverted sequence diagram, a timing diagram shows how objects interact with each other in a given timeframe
- Timing diagrams may be used to see how long each step of a process takes and find areas for improvement



# Timing Diagram: Basic concepts

- **Lifeline:** It represents a single entity, which is a part of the interaction. Ex: Following lifeline represents instances of a System and Virus:

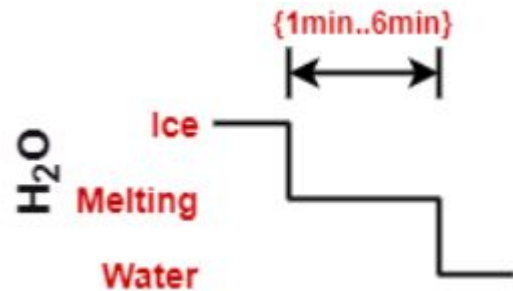


- **State or condition timeline:** represents the state of a classifier or attributes that are participating, or some testable conditions, which is a discrete value of the classifier. Ex: Following timeline shows the change in the state of virus:

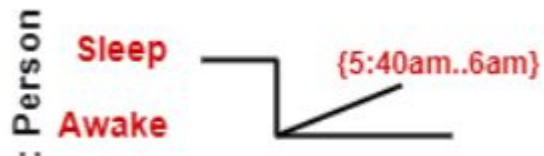


# Timing Diagram: Basic concepts

- **Duration constraint:** Duration constraint is a constraint of an interval, which refers to duration interval. It is used to determine if the constraint is satisfied for a duration or not. Ex: Following demonstrates the duration constraint ‘Ice should melt into the water in 1 to 6 minutes’:

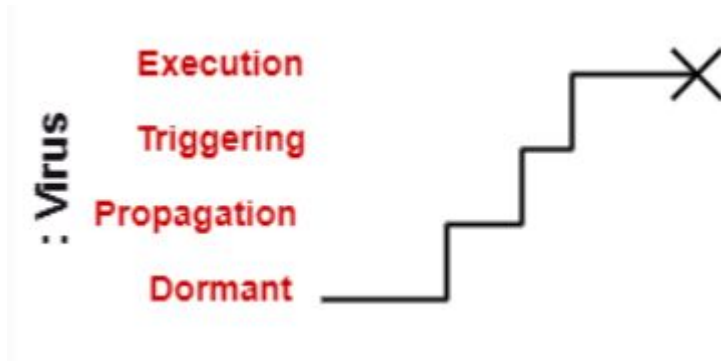


- **Time constraint:** It is an interval constraint, which refers to the time interval. As it is a time expression, it depicts if the constraint is satisfied or not. Ex: Following demonstrates the time constraint ‘A person should wakeup in between 5:40 AM and 6:00 AM:

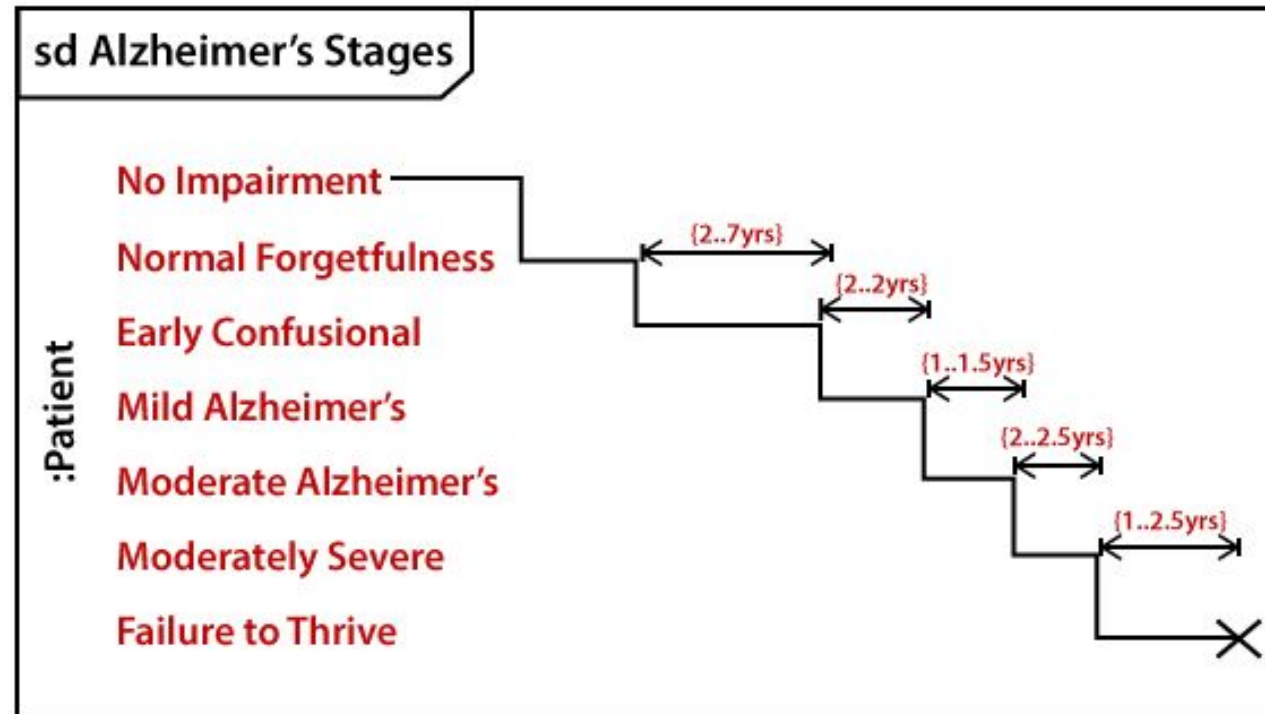


# Timing Diagram: Basic concepts

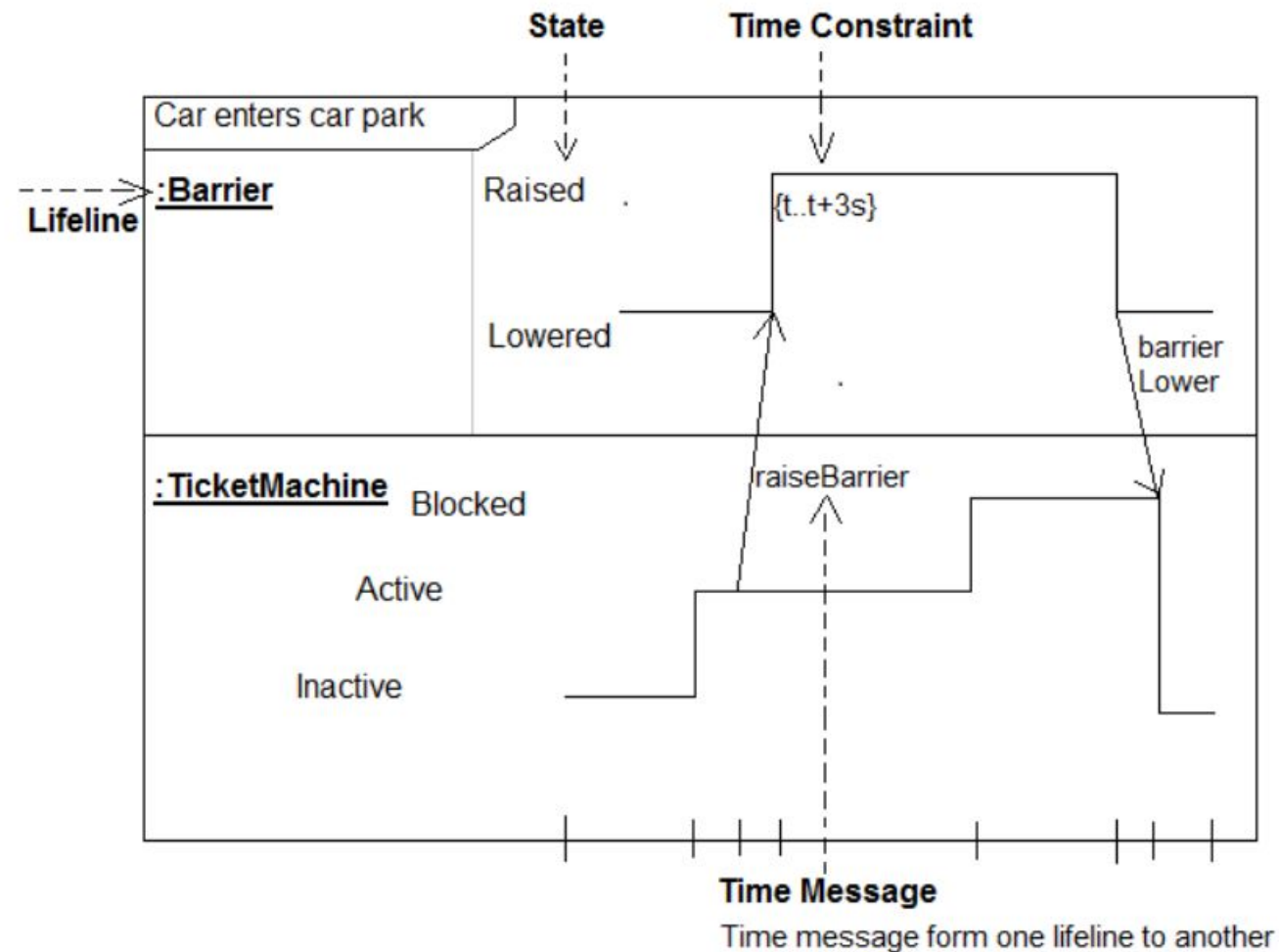
- **Destruction occurrence:** The destruction occurrence refers to the occurrence of a message that represents the destruction of an instance is defined by a lifeline. It is represented by a cross at the end of a timeline. Ex: Following demonstrates the destruction occurrence ‘Virus lifeline is terminated’:



# Timing Diagram: An Example of Alzheimer's Stages



# Timing Diagram: An Example of Car Parking



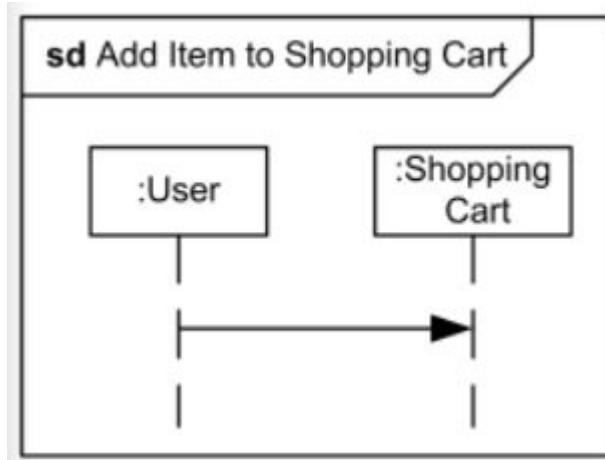
# Interaction overview Diagrams

# Interaction Overview Diagram

- Interaction overview diagrams provide a high level of abstraction of an interaction model
- It is a variant of the Activity Diagram
- It focuses on the overview of the flow of control of the interactions
- It can picture a control flow with nodes that can contain interaction diagrams which show how a set of fragments might be initiated in various scenarios
- Interaction overview diagrams focus on the overview of the flow of control where the nodes are interactions (sd) or interaction use (ref)
- The other notation elements for interaction overview diagrams are the same as for activity and sequence diagrams. These include initial, final, decision, merge, fork and join nodes

# Interaction Overview Diagram: Notations used

- **Interaction:** An interaction diagram of any kind may appear inline as an invocation action. The inline interaction diagrams may be either anonymous or named. Ex: Interaction Add Item to Shopping Cart may appear inline on some interaction overview diagram

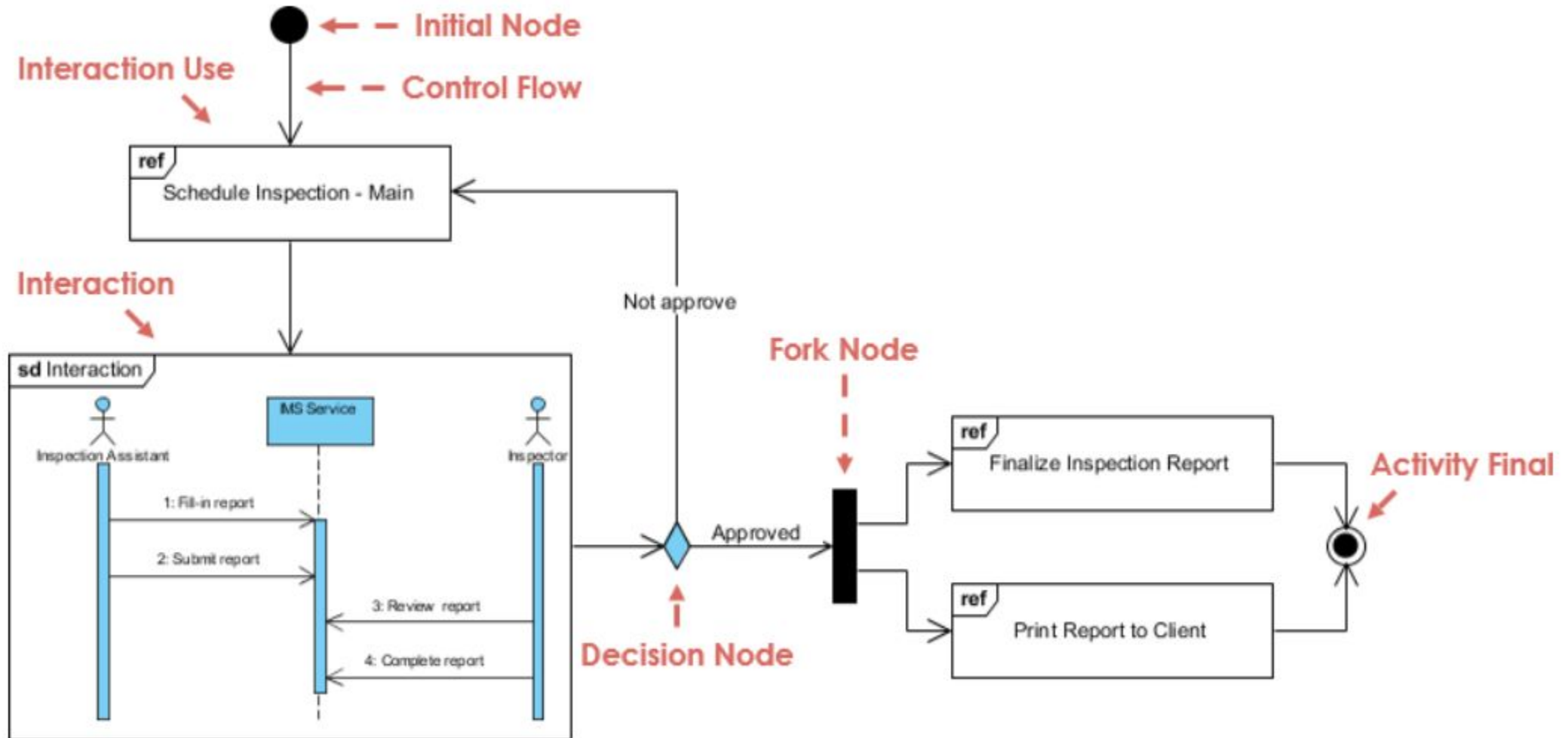


- **Interaction Use:** An interaction use may appear as an invocation action. Ex: Interaction use Add Item to Shopping Cart may appear on some interaction overview diagram



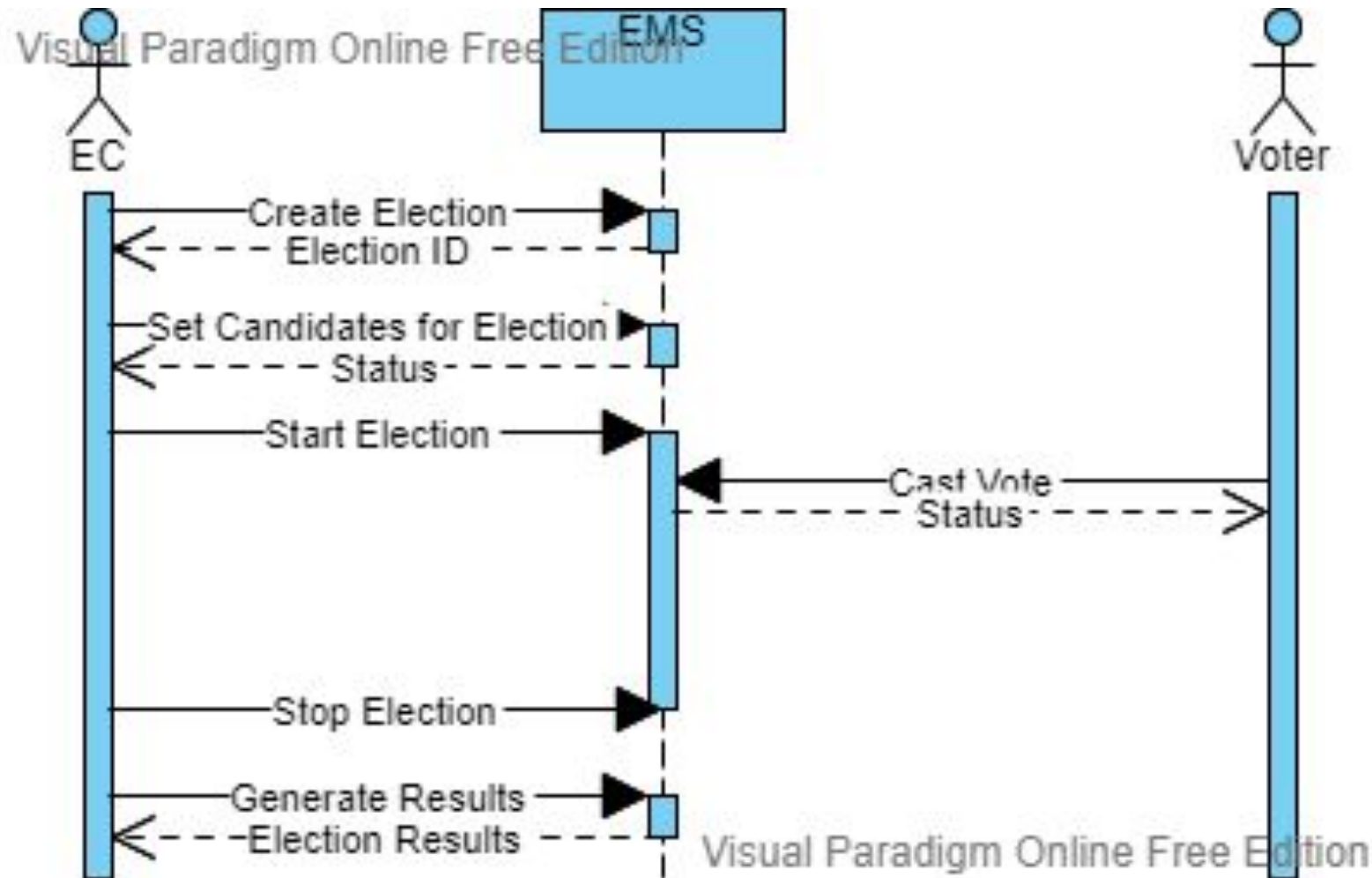


# Interaction Overview Diagram: An Example of Scheduling System

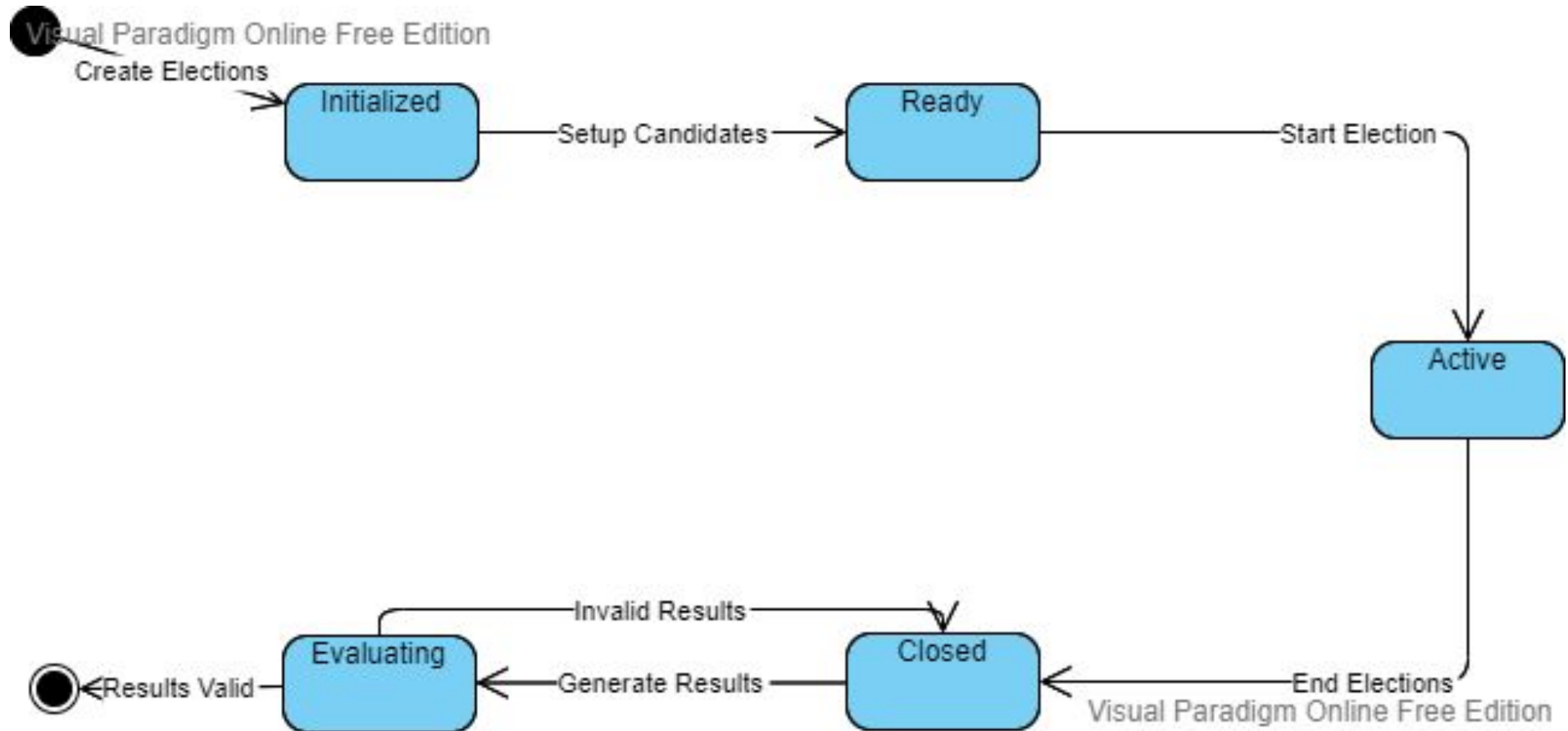


# Election Management System: An example

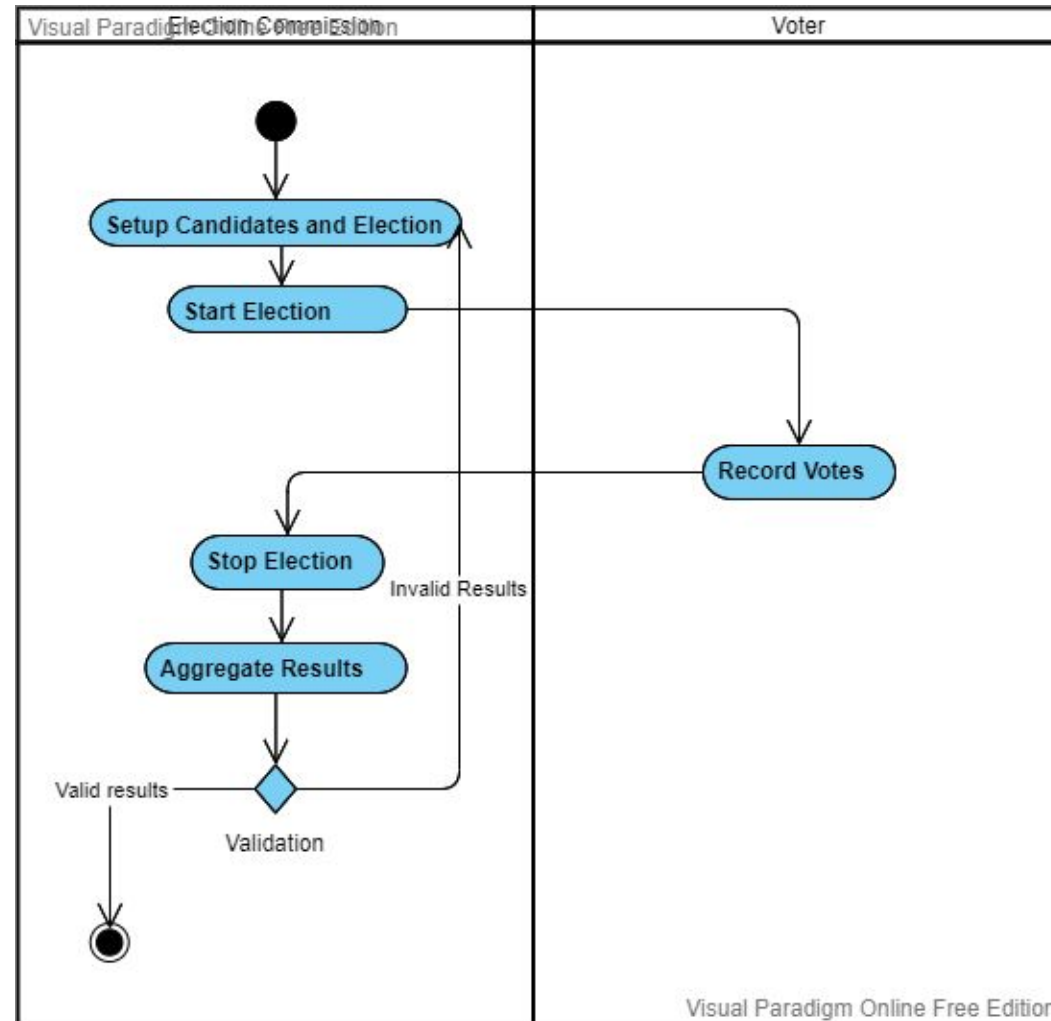
# Sequence Diagram: An Example of Election Management System



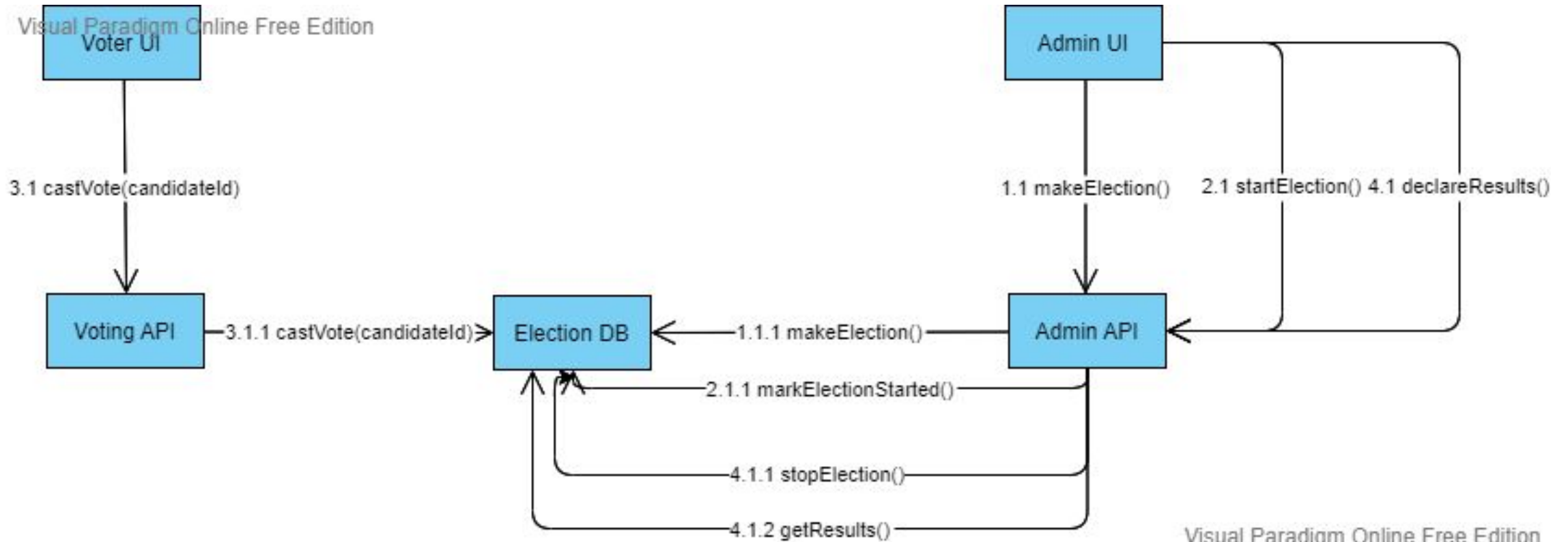
# State Machine Diagram: An Example of Election Management System



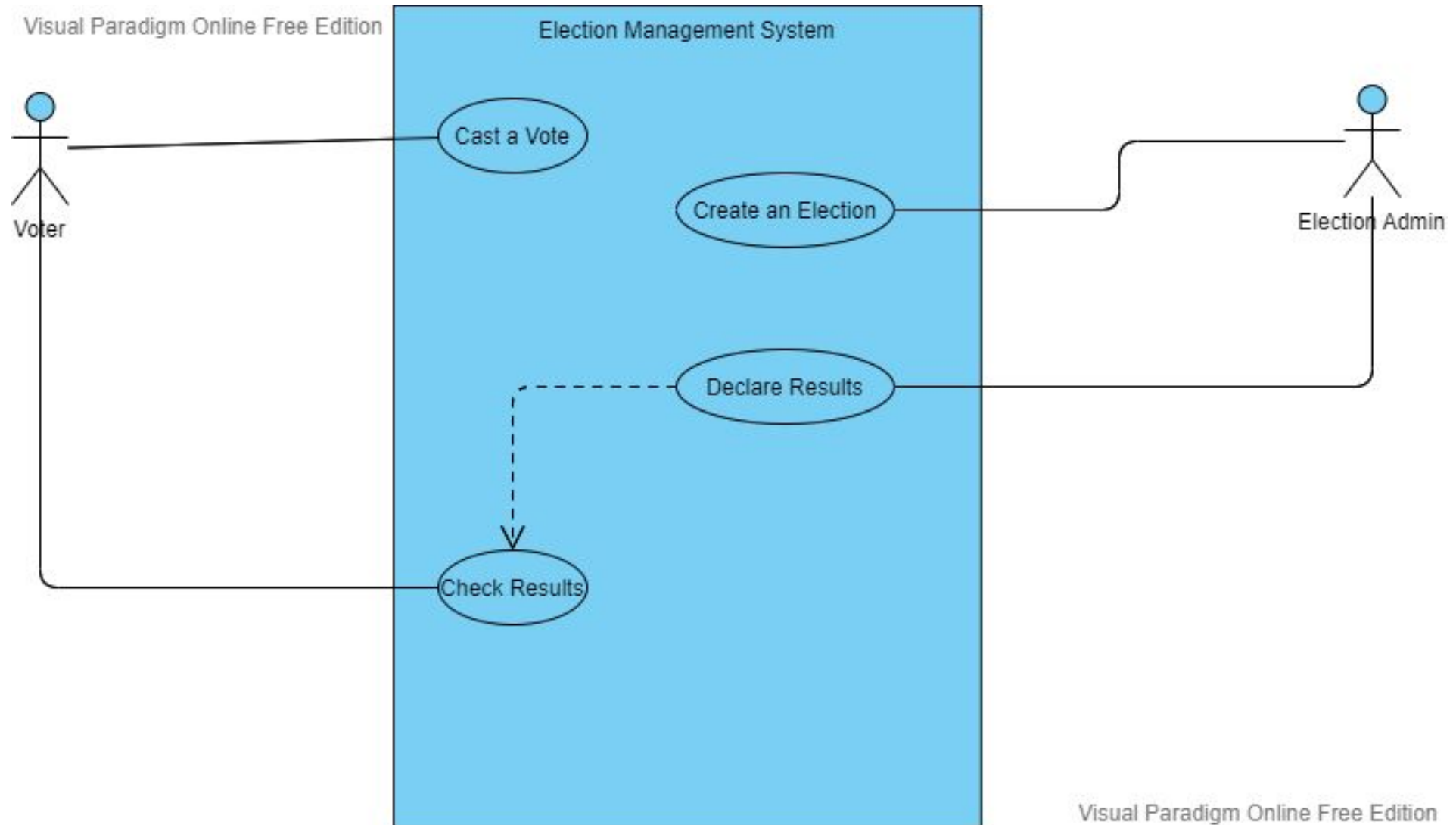
# Activity Diagram: An Example of Election Management System



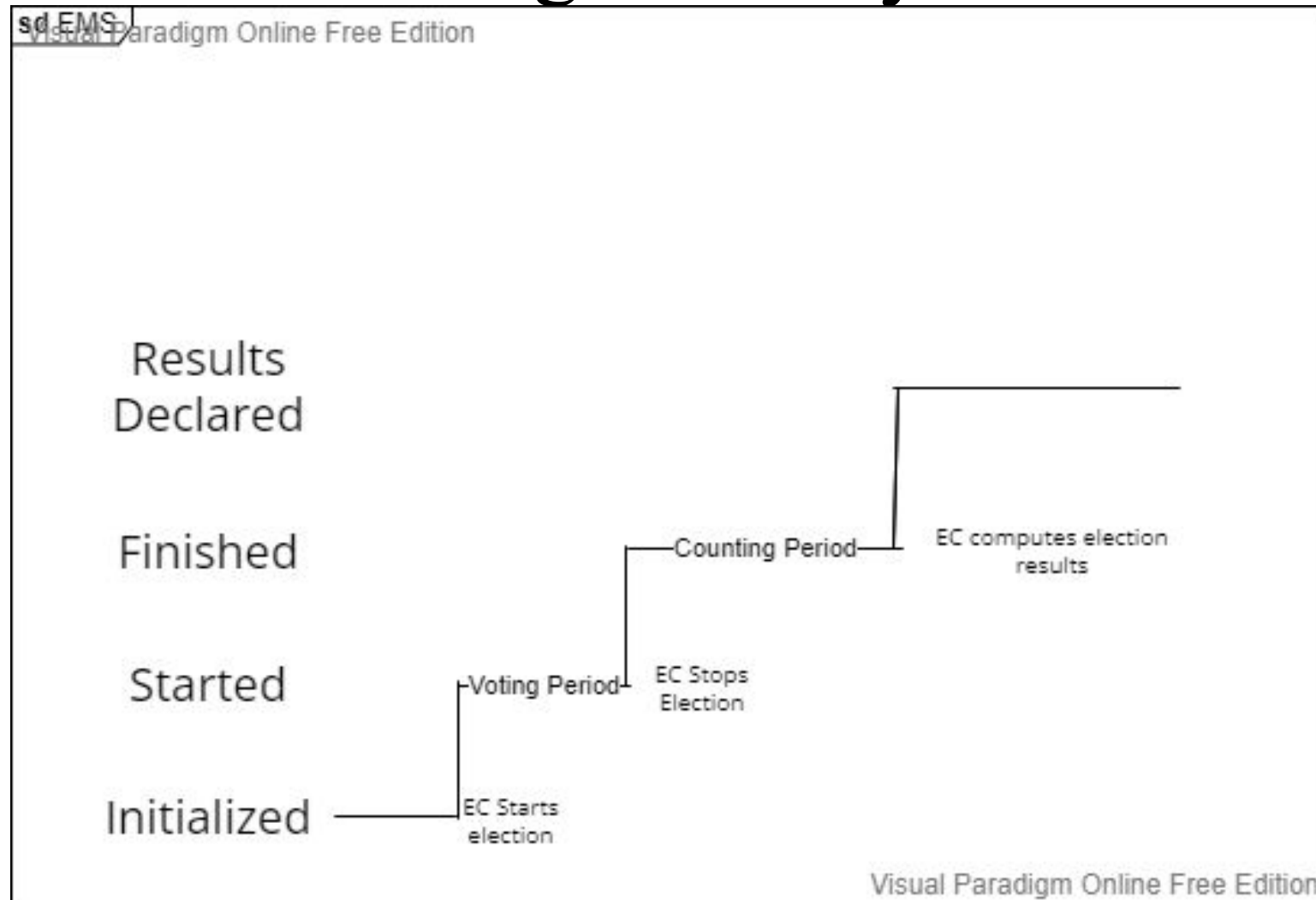
# Communication Diagram: An Example of Election Management System



# Use Case Diagram: An Example of Election Management System

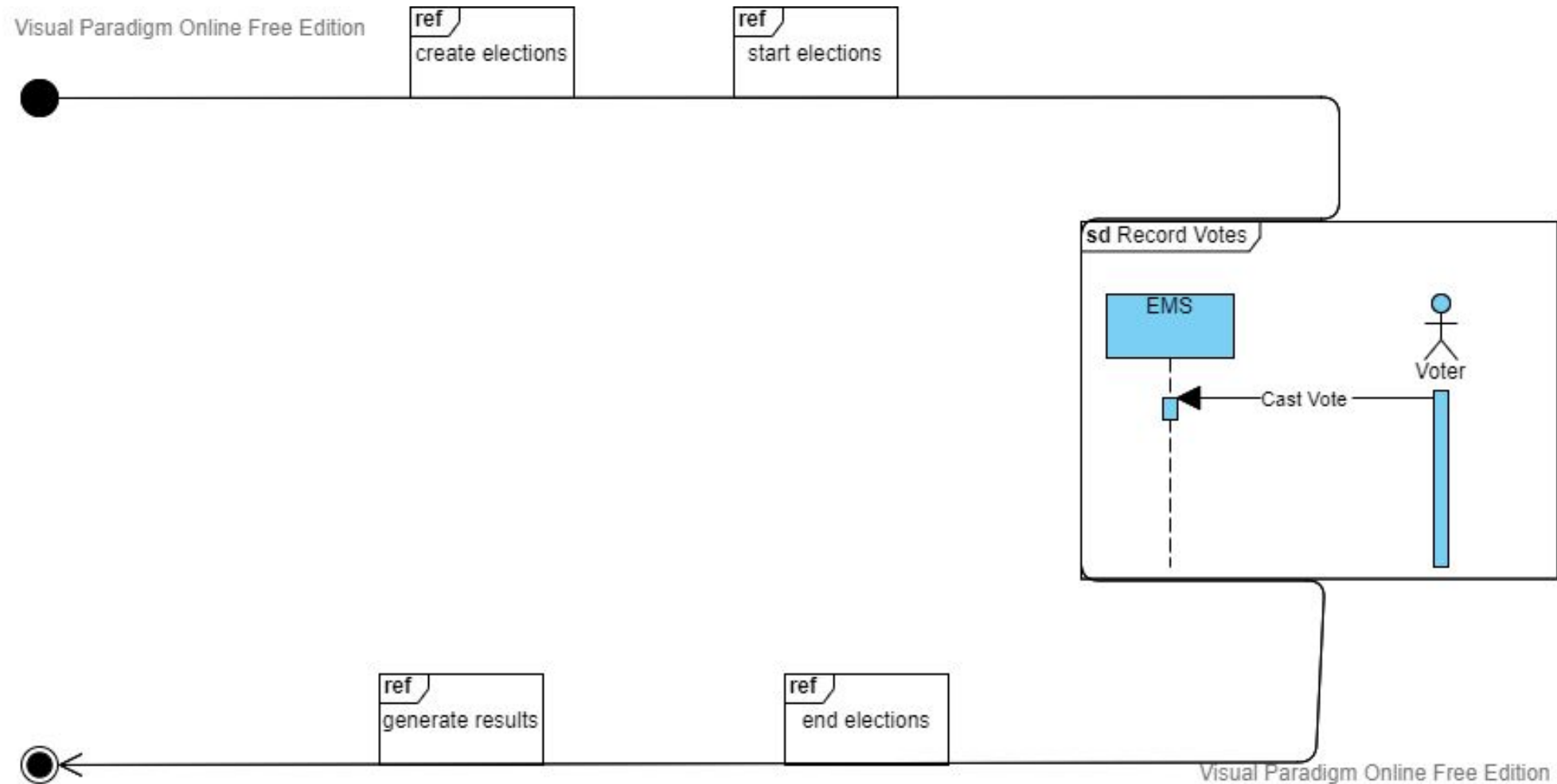


# Timing Diagram: An Example of Election Management System





# Interaction Overview Diagram: An Example of Election Management System



# VP Tool Demo

# Home Page

Visual Paradigm Online Suite

https://online.visual-paradigm.com

Visual Paradigm Online

Explore Features Resources Templates Pricing Sign up Log in

**NEW FEATURE**

## Online Flipbook Maker

Design, publish and share your beautiful flipbooks with our advanced flipbook maker online.

[Learn More](#)

Online Collage Maker

Create stunning photo collages with filters, stickers and more.

[Get Started For Free](#)

1000+ layouts

We use cookies to offer you a better experience. By visiting our website, you agree to the use of cookies as described in our [Cookie Policy](#).

[OK](#)

## Infographic Maker

Create, publish and manage all your visual content in a unified platform.

**InfoART Editor**

https://online.visual-paradigm.com/flipbook-maker/

# Dashboard

Visual Paradigm Online

Workspace

Upgrade ? 2

Project

Draft

Create New

Dashboard

My Documents

My Books

My Bookshelf

Shared with me

Import

You might want to try...

Filter ...

Suggested

Visual

Diagram

FlipBook


PhotoBook

Charts


Collage

Form


Spreadsheet




Infographics




Gift Cards




Posters




Biography Timelines



Instagram Posts



Instagram Stories

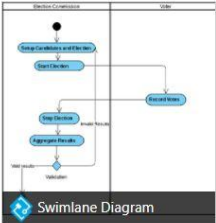


Invitations

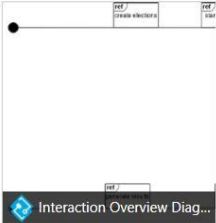
Show more types

Recent Documents

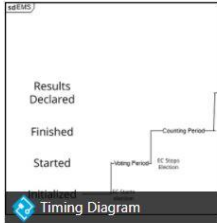
Go to My Documents



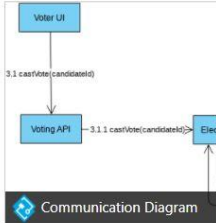
Swimlane Diagram



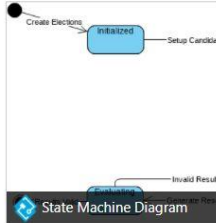
Interaction Overview Diagram



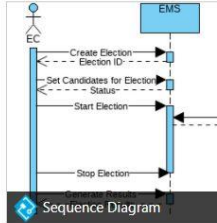
Timing Diagram




Communication Diagram



State Machine Diagram

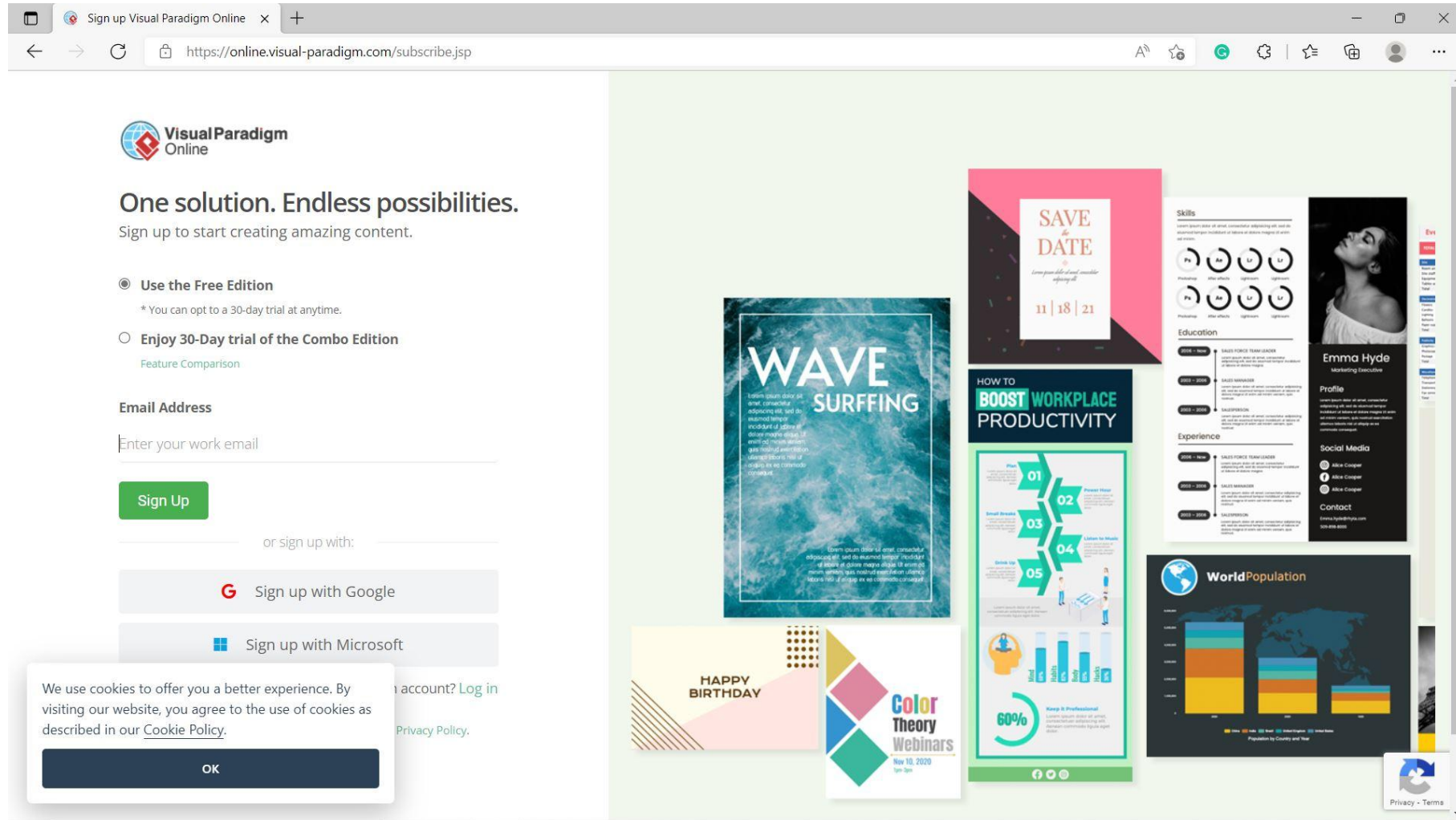


Sequence Diagram



UCD


# Signup




# Login

Log in Visual Paradigm Online

https://online.visual-paradigm.com/login.jsp

ExploreFeaturesResourcesTemplatesPricingSign upLog in



## Visual Paradigm Online

One account. All of Visual Paradigm Online

### Login to your workspace

Email Address

Password

Login

Don't have an account yet? [Sign Up](#)

or login with

Google

Microsoft

SSO

[Forgot your password?](#)

We use cookies to offer you a better experience. By visiting our website, you agree to the use of cookies as described in our [Cookie Policy](#).

OK

# My Documents

Visual Paradigm Online

Workspace

Project: Draft

Create New

Dashboard

My Documents

My Books

My Bookshelf

Shared with me

Import

Search

Swimlane Diagram

Interaction Overview Diagram

Timing Diagram

Communication Diagram

State Machine Diagram

Sequence Diagram

UCD

The screenshot displays the Visual Paradigm Online workspace for a project named 'Draft'. The interface includes a sidebar with navigation options: 'Dashboard', 'My Documents' (selected), 'My Books', 'My Bookshelf', 'Shared with me', and 'Import'. The main workspace area shows a grid of UML diagrams for an 'Election Management System'. The diagrams include: a Swimlane Diagram, an Interaction Overview Diagram, a Timing Diagram, a Communication Diagram, a State Machine Diagram, a Sequence Diagram, and a Use Case Diagram (UCD). Each diagram is represented by a thumbnail with a small icon and a title. The UCD diagram is highlighted in blue. The top of the workspace features a 'Workspace' tab, a search bar, and an 'Upgrade' button. The browser address bar shows the URL: https://online.visual-paradigm.com/w/lscjydw/drive/#infoart:proj=0&documents.

# Add New

Visual Paradigm Online

Workspace

Project: Draft

Create New

Try "infographic" or "flowchart"

Custom Size

- Timeline Diagram
- Use Case Diagram
- Infographics
- Gift Cards
- Posters
- Biography Timelines
- Instagram Posts
- Instagram Stories
- Form
- Spreadsheet

Interaction Overview Diagram

Timing Diagram

Communication Diagram

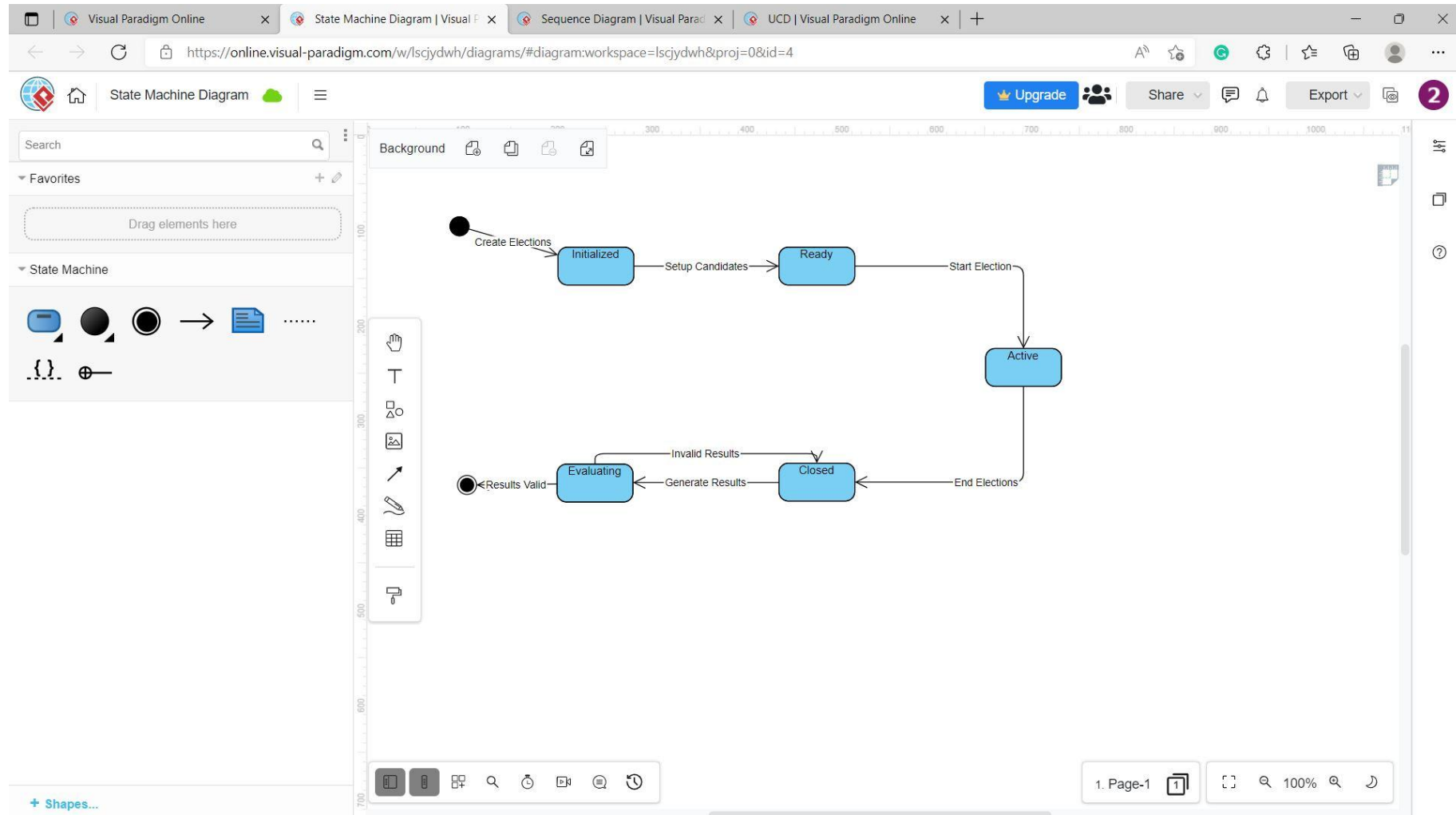
State Machine Diagram

Sequence Diagram

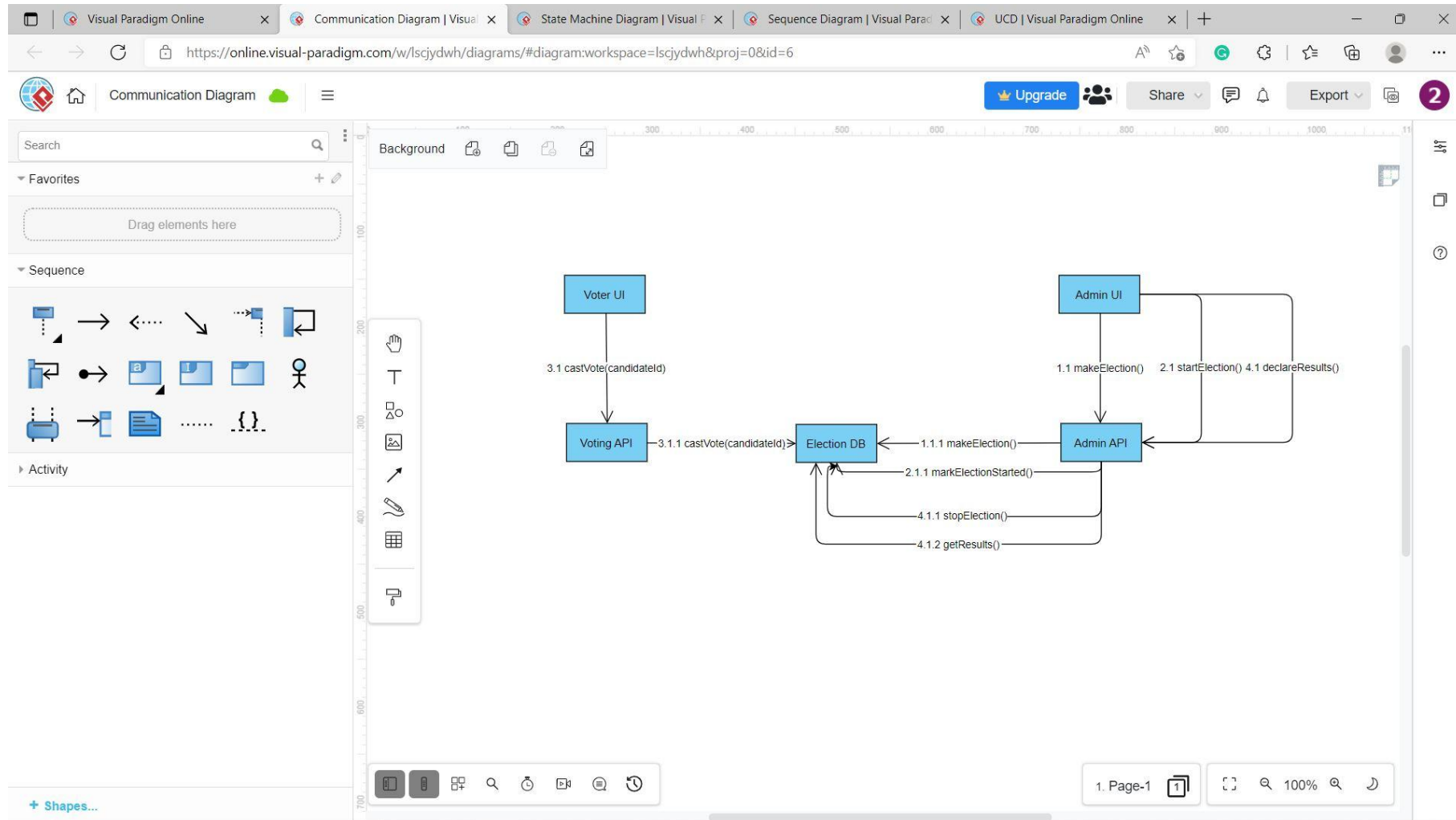
Search



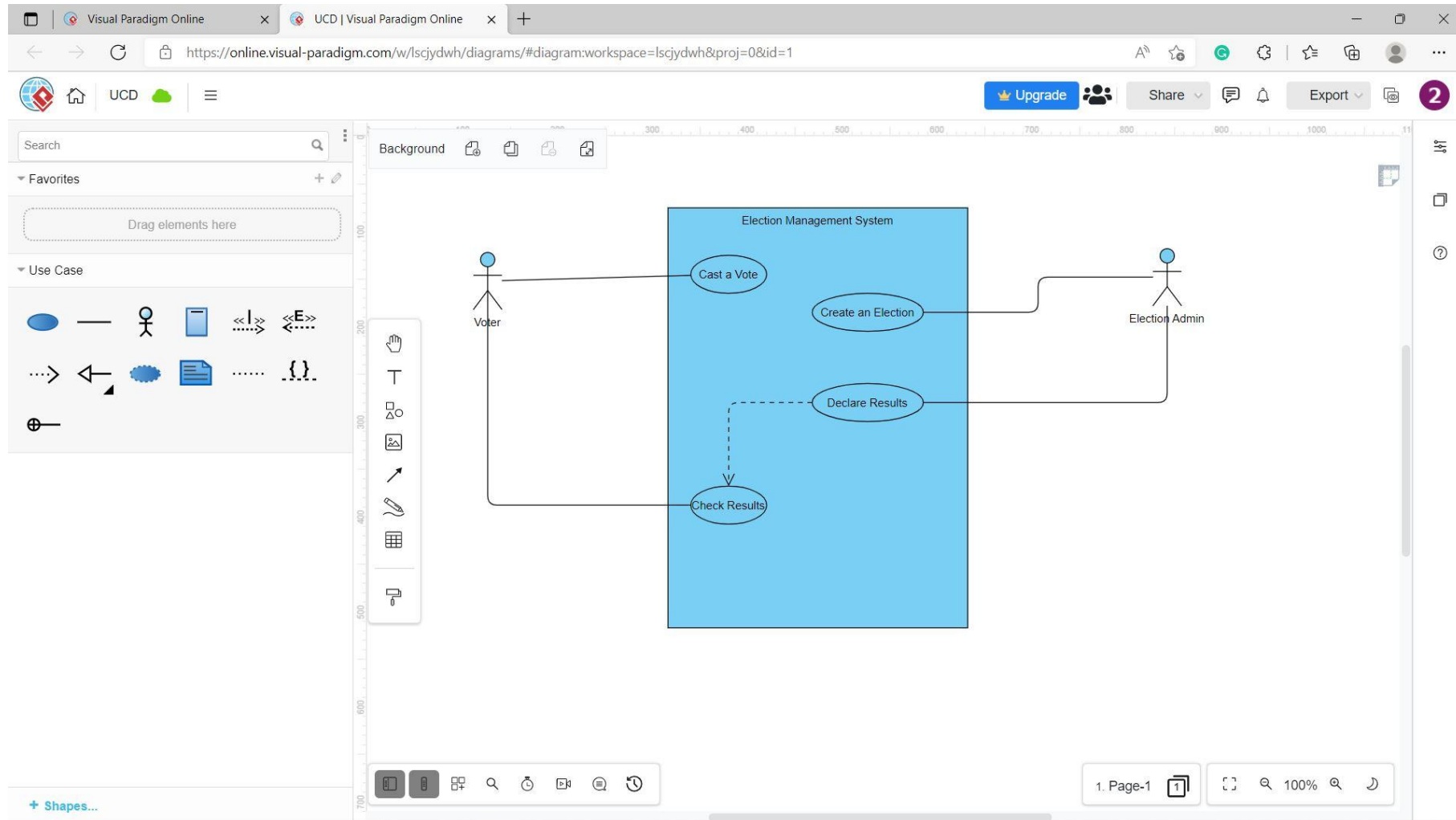
# State Machine Diagram



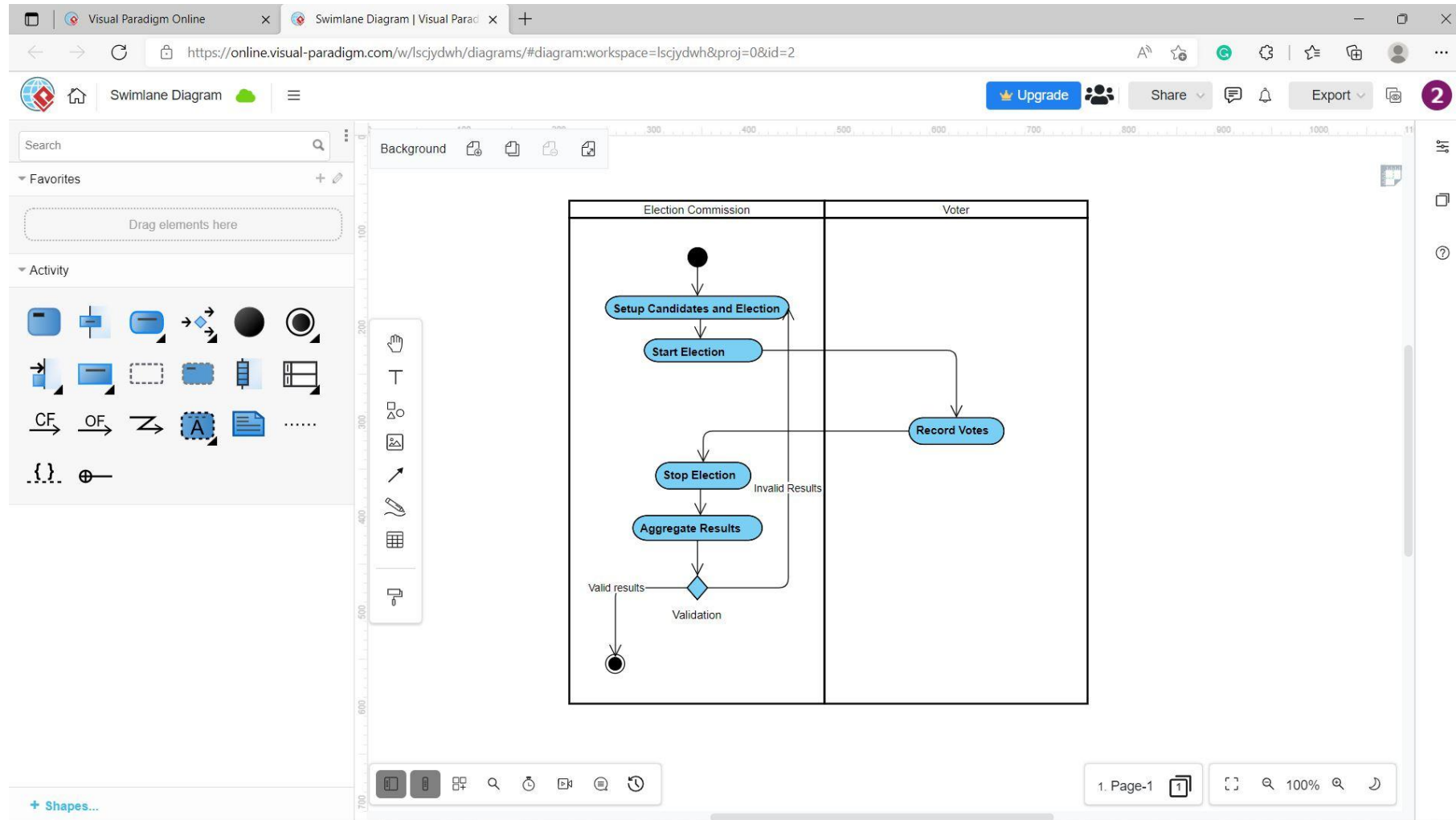
# Communication Diagram



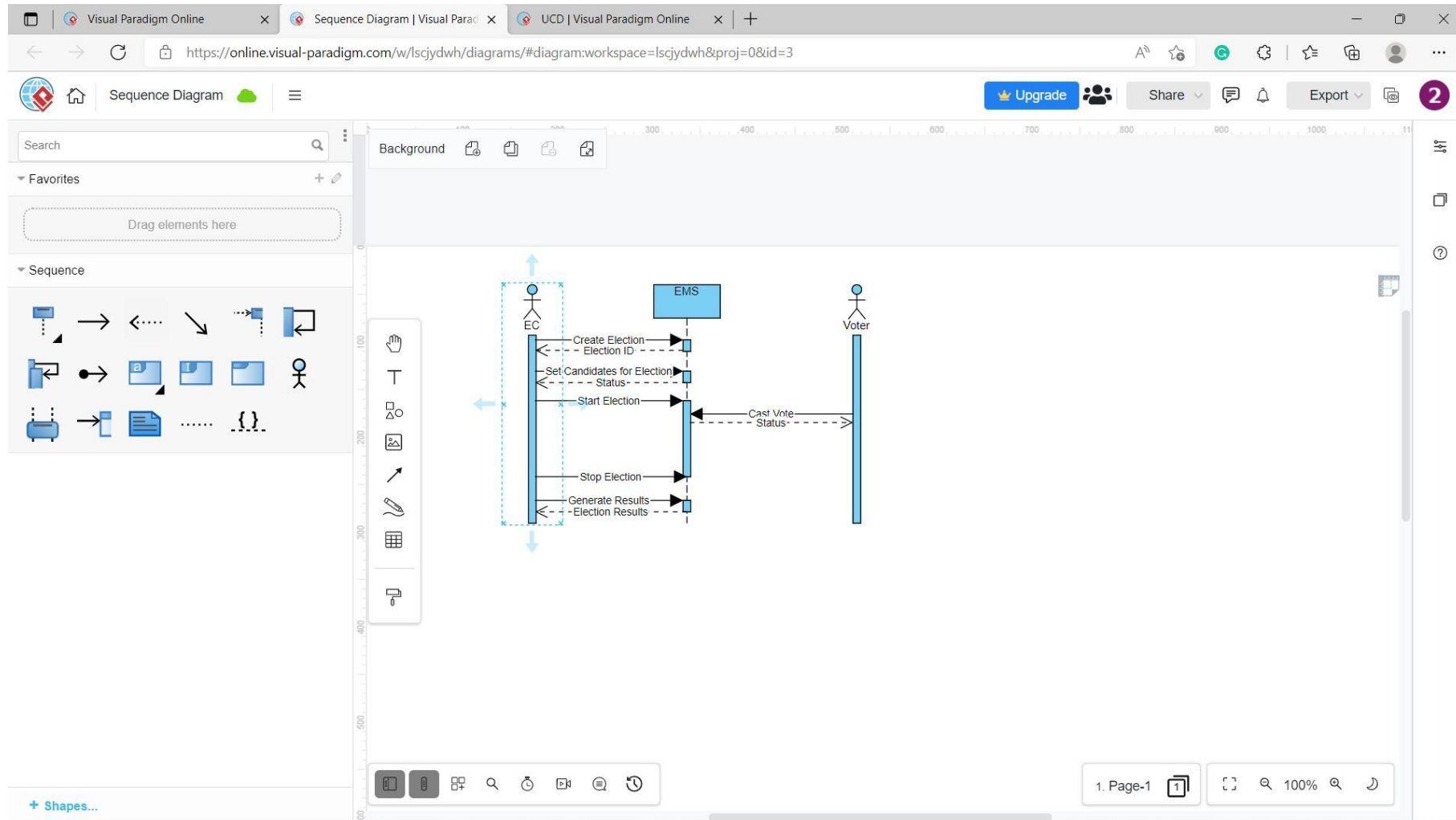
# Use Case Diagram



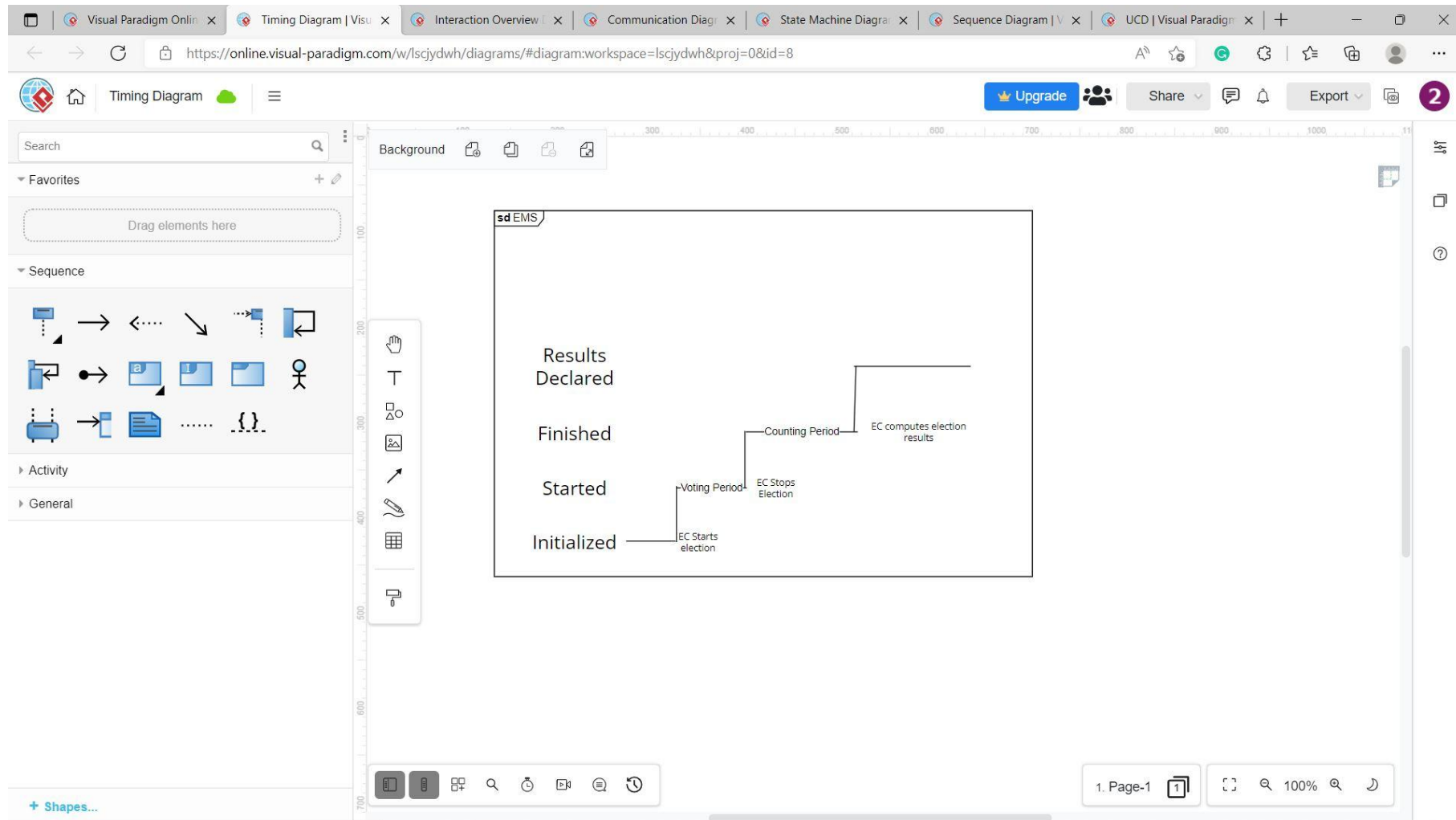
# Activity Diagram



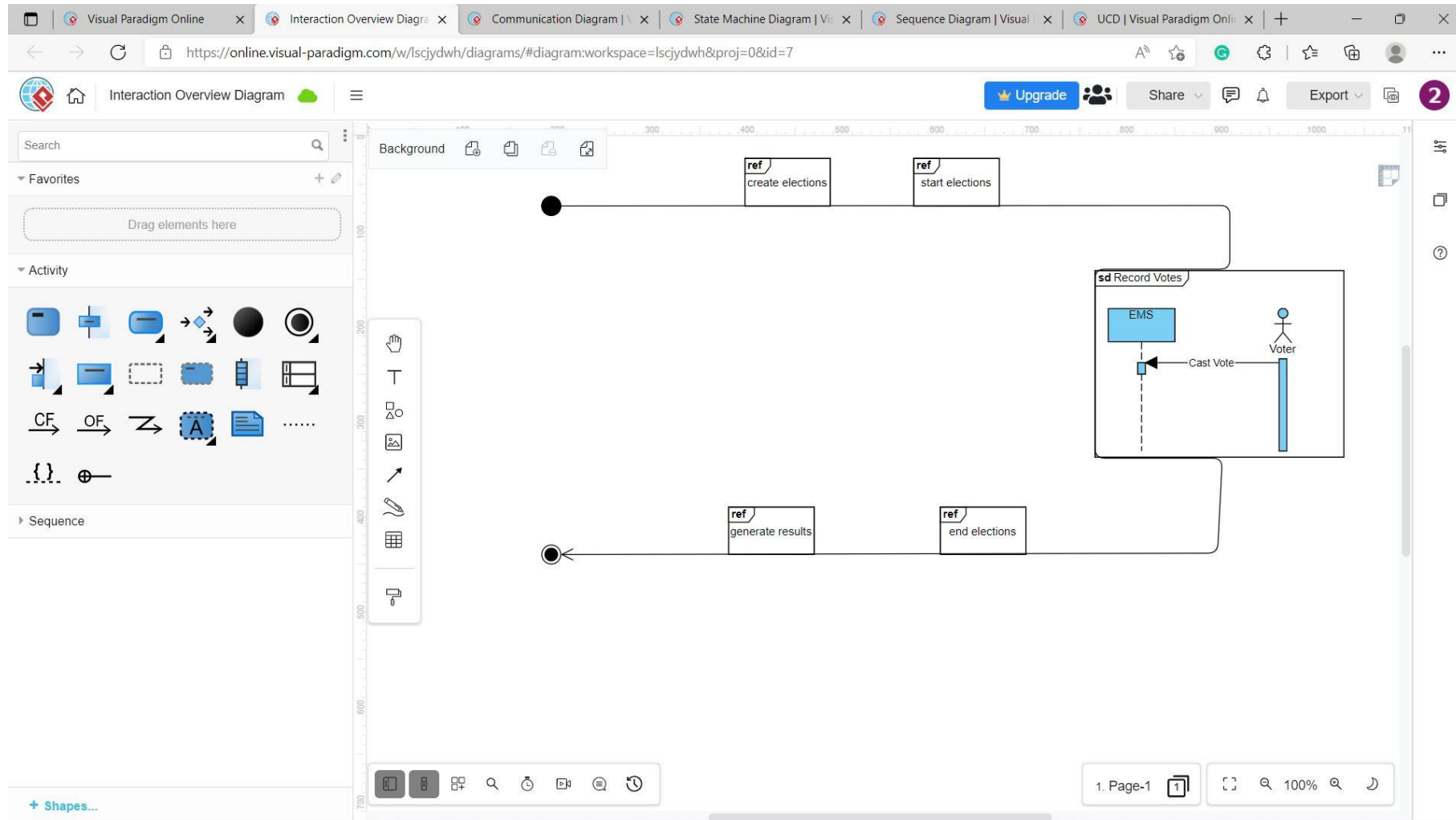
# Sequence Diagram



# Timing Diagram



# Interaction Overview Diagram





Thank You